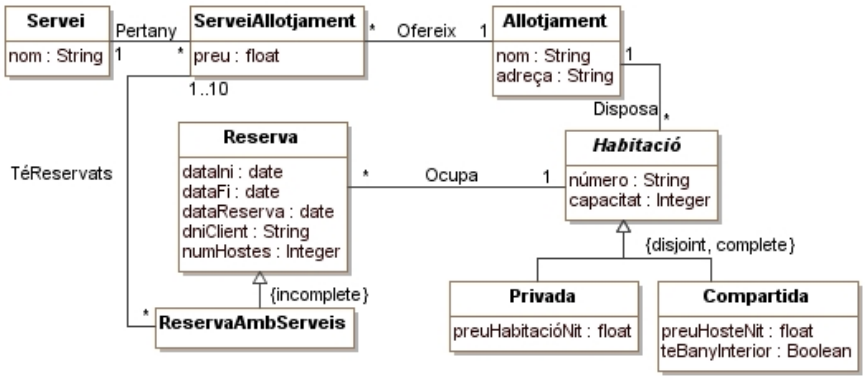
ARQUITECTURA DEL SOFTWARE

Exàmens Resolts 2on Control

Control QT17-18

1. [5,5 punts] Una cadena d'allotjaments ens havia demanat que dissenyéssim una part d'un sistema software per gestionar les reserves de les habitacions i serveis d'allotjament que ofereixen. Ara ens demanen que tornem a redissenyar el sistema utilitzant un conjunt de serveis que la cadena té disponibles. A continuació disposeu d'un fragment de l'esquema conceptual del sistema a redissenyar:

Esquema conceptual de l'especificació:



R.I. Textuals:

RI1- Claus: (Servei, nom); (ServeiAllotjament, Servei::nom + Allotjament::nom); (Allotjament, nom); (Habitació, Allotjament::nom + número); (Reserva, dniClient + dataIni)
RI2 - Una habitació no pot tenir dues reserves solapades en el temps.
RI3 - Un client no pot tenir reserves en períodes solapats.
RI4 - La *dataReserva* d'una reserva ha de ser anterior a la data d'inici de la reserva
RI5 - Les habitacions privades tenen un màxim de capacitat de dues persones.
RI6 - El *numHostes* d'una reserva ha de ser menor o igual que la capacitat de l'habitació de la reserva.
RI7 - Els serveis d'allotjament que té una reserva amb serveis han de ser oferts pel mateix allotjament de l'habitació de la reserva.
RI8 - Tots els integers i floats han de ser positius.
Altres restriccions no rellevants pel problema

A continuació disposeu del contracte de l'operació a redissenyar:

context CapaDomini :: CreaReserva (nomAllotj: String, numHab: String, dataIni: date, dataFi: date, dniClient: String, numHostes: Integer, serveis: Set(String))
pre *datesOk*: L'interval definit per *dataIni* i *dataFi* és vàlid i futur.
pre *numHostesOk*: El *numHostes* és més gran que 0.
pre *clientSenseReserves*: El client amb *dniClient* no té reserves a l'interval [*dataIni*, *dataFi*].
exc *allotjamentNoExisteix*: L'allotjament identificat per *nomAllotj* no existeix.
exc *habitacióNoExisteix*: L'habitació identificada per *nomAllotj* i *numHab* no existeix.
exc *habitacióOcupada*: L'habitació identificada per *nomAllotj* i *numHab* està ocupada en part o en tot l'interval [*dataIni*, *dataFi*].
exc *habitacióSenseCapacitat*: L'habitació identificada per *nomAllotj* i *numHab* no té capacitat per acollir el *numHostes*.
exc *serveiAllotjNoOfert*: El conjunt de serveis no és buit i algun dels serveis no és ofert per l'allotjament.
exc *moltsServeis*: El nombre de serveis del conjunt *serveis* és més gran que 10.
post *creacióReserva*: Si el conjunt de serveis és buit, es crea una reserva amb la *dataIni*, *dataFi*, *dniClient*, *numHostes*, *dataReserva* (per obtenir la data podeu usar l'operació *getDataAvui():date* invocada des d'on la necessiteu) i s'associa amb l'habitació indicada als paràmetres.
post *creacióReservaAmbServeis*: Si el conjunt de serveis no és buit, es crea una reserva amb serveis amb la *dataIni*, *dataFi*, *dniClient*, *numHostes*, *dataReserva* (podeu usar l'operació

abans indicada) i s'associa amb l'habitació i amb els serveis d'allotjament indicats als paràmetres.

Els serveis que podeu utilitzar són:

- **Servei Allotjament** (SvNomAllot): Hi ha un servei per cada allotjament. Per obtenir la referència del servei podeu utilitzar el nom de l'allotjament. Cada servei proporciona les següents operacions:

```
context SvNomAllot::infoAllotjament(): TupleType(nom:String, adreça:
String)
post result = nom i adreça de l'allotjament.

context SvNomAllot::habitacionsAllot(): Set(TupleType(número:String,
capacitat: Integer, tipus:String, preuHabitacióNit[0..1]:float,
preuHosteNit[0..1]:float, teBanyInterior[0..1]:Boolean))
post result = conjunt de les dades de les habitacions de
l'allotjament. Per a cada habitació es retorna el número, capacitat i
tipus de l'habitació. Si el tipus de l'habitació és Privada es retorna
també el preuHabitacióNit i si és Compartida el preuHosteNit i si té
banyInterior.

context SvNomAllot::serveisAllot (): Set(TupleType(nom:String, preu:
float))
post result = conjunt de les dades dels serveis que ofereix
l'allotjament. Per a cada servei ofert es retorna el nom i el preu.
```

- **Servei Reserva** (SvRes): Enregistra reserves d'habitacions. Fixeu-vos que el servei no diferencia entre les reserves sense i amb serveis. El servei proporciona les següents operacions.

```
context SvRes::getReservesxHabitació (nomAllot:String, numHab:String):
Set( TupleType(dataIni: date, dataFi: date, dniClient: String,
numHostes: Integer))
post result = conjunt de les dades de les reserves de l'habitació
indicada al paràmetre. Per a cada reserva es retorna la dataIni,
dataFi, dniClient, i numHostes.

context SvRes::altaReserva (nomAllot: String, numHab: String, dataIni:
date, dataFi: date, dniClient: String, numHostes: Integer)
exc reservaExisteix: la reserva indicada als paràmetres ja existeix.
exc habitacióOcupada: l'habitació identificada per nomAllotj i numHab
està ocupada en part o en tot l'interval [dataIni, dataFi].
exc clientAmbReservesSolapades: el client amb dniClient ja té reserves
a l'interval [dataIni, dataFi].
post creacióReserva: es dóna d'alta una reserva amb les dades
indicades als paràmetres.
```

Es demana:

a) [1,5 punts] Diagrama de classes del paquet Domain Model de la capa de domini del sistema. Heu d'incloure els atributs (però no les operacions). **Expliqueu** quina informació queda sota la responsabilitat del sistema local i quina sota la responsabilitat dels serveis.

b) [4 punts] Diagrama de seqüència de l'operació *CreaReserva*. **Com a criteris de disseny volem minimitzar el nombre d'invocacions remotes mantenint els criteris habituals de reusabilitat, canviabilitat i minimitzant la redundància de les dades.**

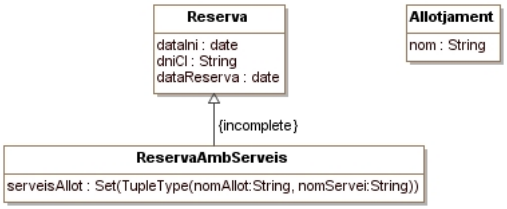
2. [0,5 punts] Explica **clarament** com pot garantir la Capa de Presentació o les pantalles la precondició *clientSenseReserves* del contracte anterior. No cal que feu el mapa navegacional.

3. [4 punts] Considerant el diagrama de classes i restriccions d'integritat donats a l'exercici 1.

- **a)** [1.5 punts] Disseny de la base de dades usant Class Table Inheritance per a la jerarquia d'habitacions i Concrete Table Inheritance per a la jerarquia de reserves. Useu restriccions de clau i altres restriccions de columna i de taula sempre que es pugui **(No es poden usar triggers**).
- **b)** [2.5 punts] Disseny de l'operació *CreaReserva* usant **el patró Transaction Script amb Row Data Gateway** (Pasarel.la Fila) (**sense operacions arbitràries de consulta**). No es poden usar els *SvNomAllotjament* i *SvRes*. Mostreu clarament en quin moment es llancen les excepcions indicades en l'operació.

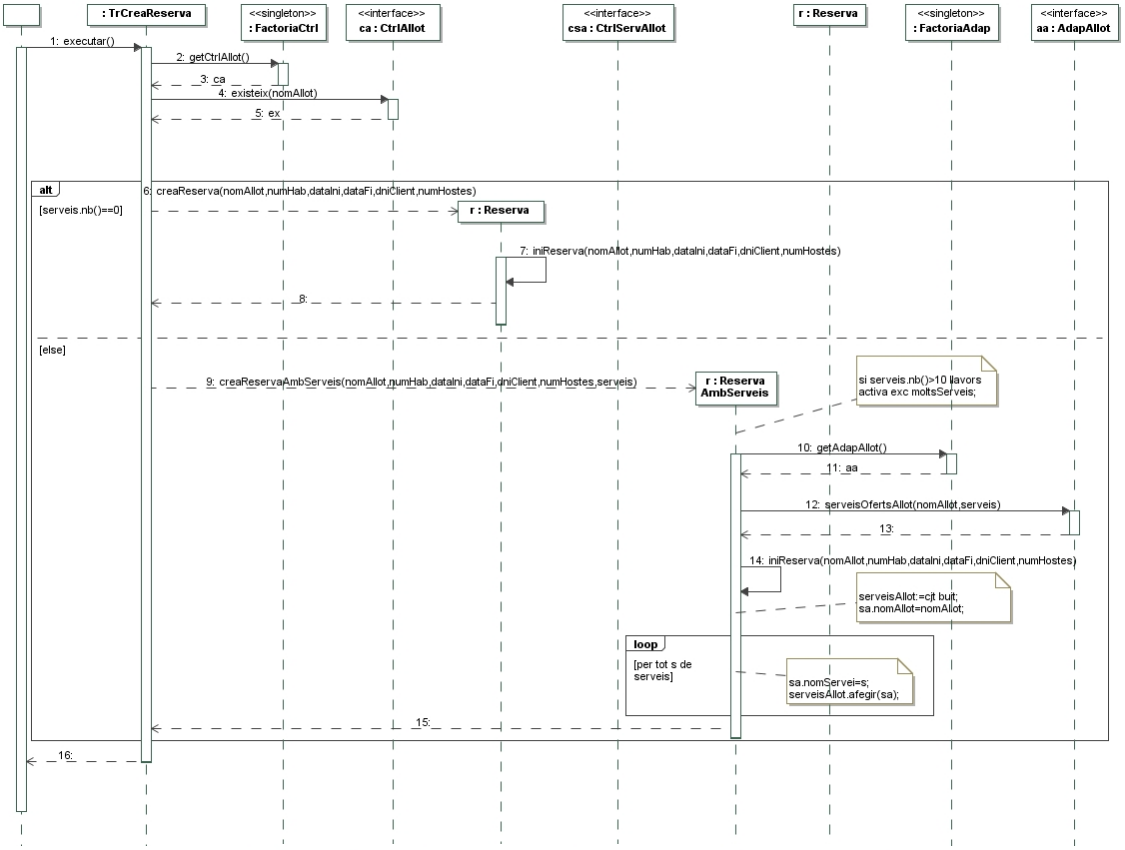
Solució Control QT17-18

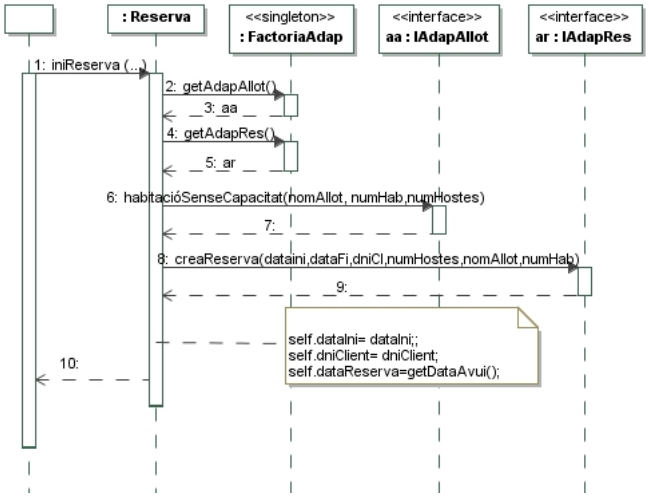
1. a)

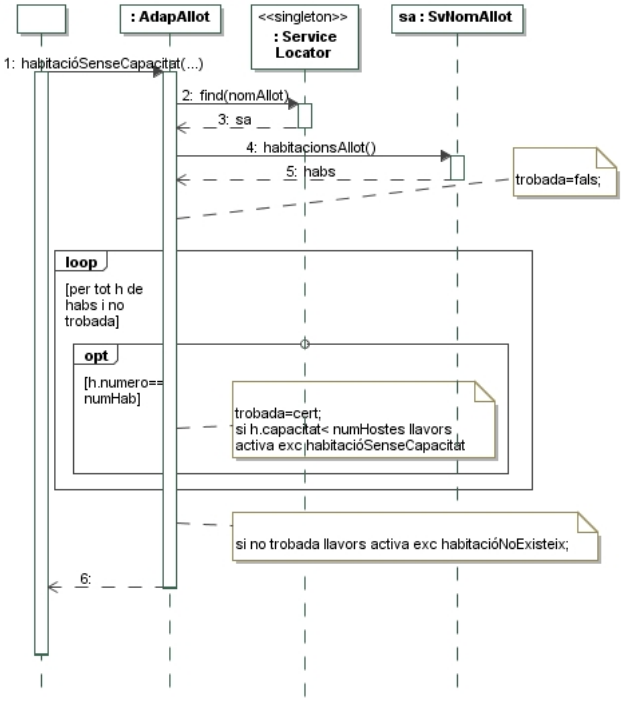


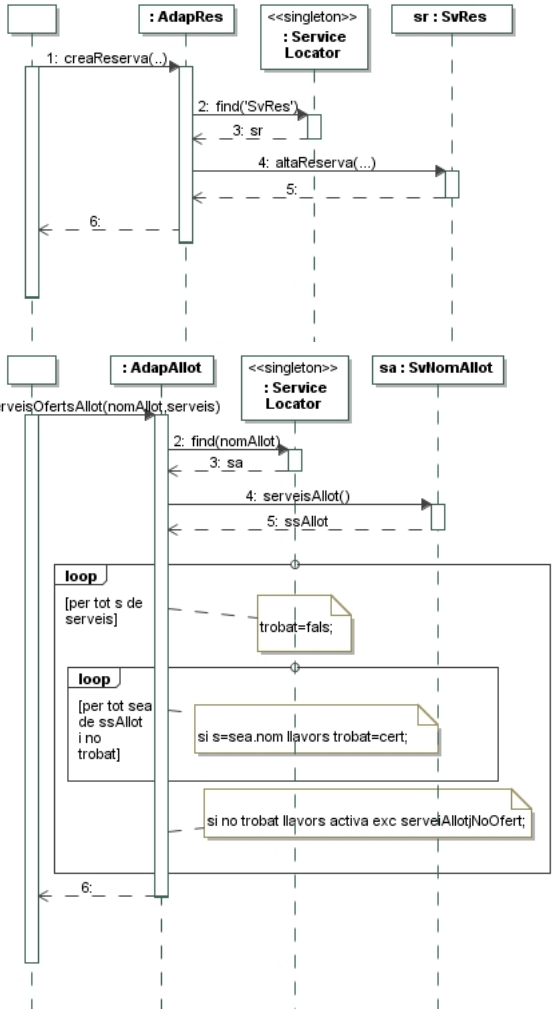
Al sistema local tindrem de totes les reserves i reserves amb serveis la informació que no guarden els serveis (bàsicament la dataReserva per a totes les reserves i el serveis d'allotjament pel cas de les reserves amb serveis juntament amb la clau de les reserves). A més s'emmagatzema els diferents allotjaments de la cadena d'allotjaments.

b)


: TrCreaReserva
<<singleton>>
: FactoriaCtrl
<<interface>>
ca : CtrlAllot
<<interface>>
csa : CtrlServAllot
r : Reserva
<<singleton>>
: FactoriaAdap
<<interface>>
aa : AdapAllot
1: executar()
2: getCtrlAllot()
3: ca
4: existeix(nomAllot)
5: ex
alt
[serveis.nb()==0]
6: creaReserva(nomAllot,numHab,datalni,dataFi,dniClient,numHostes)
r : Reserva
7: iniReserva(nomAllot,numHab,datalni,dataFi,dniClient,numHostes)
8:
[else]
9: creaReservaAmbServeis(nomAllot,numHab,datalni,dataFi,dniClient,numHostes,serveis)
r : Reserva
AmbServeis
si serveis.nb()>10 llavors
activa exc moltsServeis;
10: getAdapAllot()
11: aa
12: serveisOfertsAllot(nomAllot,serveis)
13:
14: iniReserva(nomAllot,numHab,datalni,dataFi,dniClient,numHostes)
serveisAllot:=cjt buit;
sa.nomAllot=nomAllot;
loop
[per tot s de
serveis]
sa.nomServei=s;
serveisAllot.afegir(sa);
15:
16:


: Reserva
<<singleton>>
: FactoriaAdap
<<interface>>
aa : IAdapAllot
<<interface>>
ar : IAdapRes
1: iniReserva (...)
2: getAdapAllot()
3: aa
4: getAdapRes()
5: ar
6: habtacióSenseCapacitat(nomAllot, numHab,numHostes)
7:
8: creaReserva(dataini,dataFi,dniCl,numHostes,nomAllot,numHab)
9:
self.datalni= datalni;;
self.dniClient= dniClient;
self.dataReserva=getDataAvui();
10:

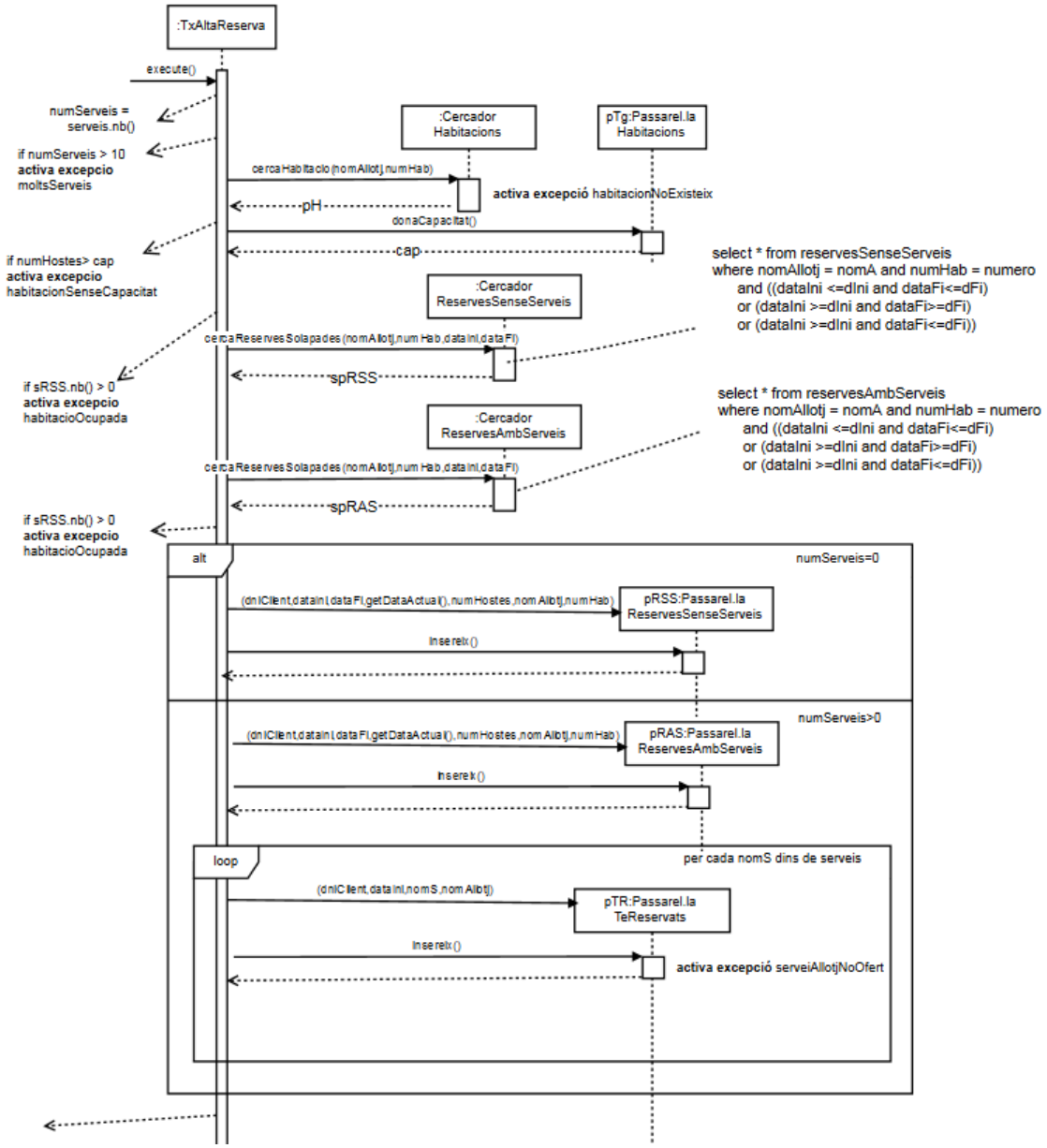

: AdapAllot
<<singleton>>
: Service Locator
sa : SvNomAllot
1: habitacióSenseCapacitat(...)
2: find(nomAllot)
3: sa
4: habitacionsAllot()
5: habs
trobada=fals;
loop
[per tot h de habs i no trobada]
opt
[h.numero== numHab]
trobada=cert;
si h.capacitat< numHostes llavors activa exc habitacióSenseCapacitat
si no trobada llavors activa exc habitacióNoExisteix;
6:


: AdapRes
<<singleton>>
: Service Locator
sr : SvRes
1: creaReserva(..)
2: find('SvRes')
3: sr
4: altaReserva(...)
5:
6:
: AdapAllot
<<singleton>>
: Service Locator
sa : SvNomAllot
1: serveisOfertsAllot(nomAllot,serveis)
2: find(nomAllot)
3: sa
4: serveisAllot()
5: ssAllot
loop
[per tot s de serveis]
trobat=fals;
loop
[per tot sea de ssAllot i no trobat]
si s=sea.nom llavors trobat=cert;
si no trobat llavors activa exc serveiAllotjNoOfert;
6:

2. La Capa de Presentació pot garantir la precondició:
pre *clientSenseReserves*: El client amb *dniClient* no té reserves a l'interval [*dataIni*, *dataFi*].
En una primera interacció s'hauran introduït les dates (que s'haurà comprovat que són vàlides) i un client d'un desplegable de clients que no tenen reserves a l'interval introduït.

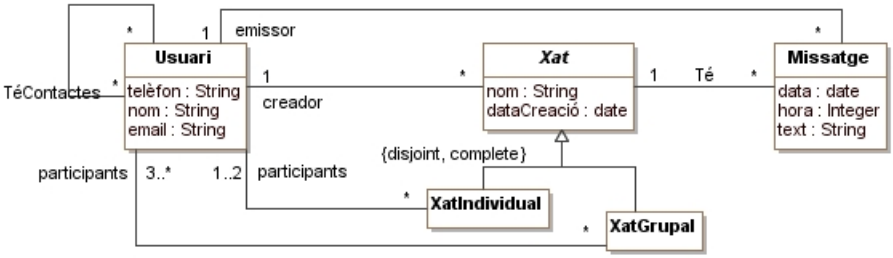
3.

```
serveis(nomS)

allotjaments(nomA,adreça)

serveisallotjament(nomS,nomA,preu)
    {nomS} referencia serveis
    {nomA} referencia allotjaments

habitacions(nomA,numero,capacitat)

habitacionsPrivades(nomA, numero, preuHabNit)
    {nomA,numero} referencia habitacions

habitacionsCompartides(nomA, numero, capacitat, preuHosteNit, teBany)
    {nomA, numero} referencia habitacions

reservesAmbServeis(dniClient, dIni, dFi, dRes, numHostes, nomA,numero)
    {nomA, numero} referencia habitacions NOT NULL

reservesSenseServeis(dniClient, dIni, dFi, dRes, numHostes, nomA,numero)
       {nomA, numero} referencia habitacions NOT NULL

teReservats(dniClient, dataIni, nomS, nomA)
    {dniClient, dataIni} referencia reservesAmbServeis
    {nomS,nomA} referencia serveisAllotjament

RI1 - PKs de les taules
RI2 - no es pot implementar a l'esquema
RI3 - no es pot implementar a l'esquema
RI4 - taula reservesAmbServeis check(dIni>dRes)
    taula reservesSenseServeis check(dIni>dRes)
RI5 - no es pot implementar a l'esquema
RI6 - no es pot implementar a l'esquema
RI7 - no es pot implementar a l'esquema, ho garantirà la transacció que donarà
d'alta la reserva ja que només hi haurà un nom d'allotjament i no dos (capa de
presentació)
RI8 - preu, capacitat, preuHabNit, preuHosteNit, numHostes check(atribut >0)
```


:TxAltaReserva
execute()
numServeis = serveis.nb()
if numServeis > 10
activa excepcio
moltsServeis
:Cercador Habitacions
pTg:Passarel.la Habitacions
cercaHabitacio(nomAllotj,numHab)
activa excepció habitacionNoExisteix
pH
donaCapacitat()
cap
if numHostes> cap
activa excepcio
habitacionSenseCapacitat
select * from reservesSenseServeis
where nomAllotj = nomA and numHab = numero
and ((dataIni <=dIni and dataFi<=dFi)
or (dataIni >=dIni and dataFi>=dFi)
or (dataIni >=dIni and dataFi<=dFi))
:Cercador ReservesSenseServeis
cercaReservesSolapades(nomAllotj,numHab,dataIni,dataFi)
spRSS
if sRSS.nb() > 0
activa excepcio
habitacioOcupada
select * from reservesAmbServeis
where nomAllotj = nomA and numHab = numero
and ((dataIni <=dIni and dataFi<=dFi)
or (dataIni >=dIni and dataFi>=dFi)
or (dataIni >=dIni and dataFi<=dFi))
:Cercador ReservesAmbServeis
cercaReservesSolapades(nomAllotj,numHab,dataIni,dataFi)
spRAS
if sRSS.nb() > 0
activa excepcio
habitacioOcupada
alt
numServeis=0
(dniClient,dataIni,dataFi,getDataActual(),numHostes,nomAllotj,numHab)
pRSS:Passarel.la ReservesSenseServeis
inserir()
numServeis>0
(dniClient,dataIni,dataFi,getDataActual(),numHostes,nomAllotj,numHab)
pRAS:Passarel.la ReservesAmbServeis
inserir()
loop
per cada nomS dins de serveis
(dniClient,dataIni,nomS,nomAllotj)
pTR:Passarel.la TeReservats
inserir()
activa excepció serveiAllotjNoOfert

Control QP17-18

1. [5 punts] Ens demanen redissenyar el software de missatgeria que vàrem dissenyar al primer control utilitzant un conjunt de serveis disponibles. A continuació disposeu d'un fragment de l'esquema conceptual del sistema a redissenyar.
Esquema conceptual de l'especificació:



R.I. Textuals:
RI1 - Claus: (Usuari, telèfon); (Xat, nom + Usuari::telèfon); (Missatge, Xat::(nom +Usuari creador::telèfon) + data + hora);
RI2 - El creador d'un xat ha de ser un dels seus participants.
RI3 - L'emissor d'un missatge d'un xat ha de ser un dels participants del xat.
RI4 - La data d'un missatge ha de ser posterior a la data de la creació del xat.
RI5 - Els participants d'un xat, sense incloure el creador del xat, han de ser contactes de l'usuari creador.
- Altres restriccions no rellevants pel problema

A continuació disposeu del contracte de l'operació a dissenyar:

context CapaDomini:: *AltaXatGrupal* (nomXat: String, usuariCreador: String, participants: Set(String))
exc *xatGrupalExisteix:* el xat identificat als paràmetres existeix.
exc *mínimParticipantsNoOk:*.el nombre de participants del conjunt *participants* és menor que 3.
exc *creadorNoParticipant:* l'*usuariCreador* no hi és al conjunt *participants*.
exc *algunParticipantNoContacte:* algun dels participants, sense incloure l'*usuariCreador*, no és un contacte de l'*usuariCreador* .
post *altaXatGrupal*: es dóna d'alta el xat grupal amb data de creació la data d'avui (podeu utilitzar l'operació del sistema *getDataAvui(): date*) i s'inicialitzen les seves dades.

Els serveis que podeu utilitzar són:

- **Servei Xats** (SvXat): Enregistra els xats i els seus missatges. El servei proporciona les següents operacions.

```
context SvXat::enviarMissatgeAXat (nomXat:String,
usuariCreador:String, data:date, hora: Integer, emissor: String, text:
String)
exc xatNoExisteix: el xat indicat als paràmetres no existeix.
exc missatgeExisteix: el missatge indicat als paràmetres existeix.
exc emissorNoParticipant: l'emissor no és el creador o participant del
xat.
post result = crea el missatge i l'assigna al xat i a l'emissor.

context SvXat::nouXat (nomXat:String, usuariCreador:String,
tipusXat:String, participants: Set(String), data Creació: date)
exc xatExisteix: el xat indicat als paràmetres ja existeix.
```

exc *mínimParticipantsnoOk:* el tipus de xat és individual i participants està buit o el tipus de xat és grupal i participants té menys de tres elements.

exc *màximParticipantsnoOk:* el tipus de xat és individual i participants té més de dos elements.

exc *creadorNoParticipant:* l'usuari creador no hi és al conjunt participants.

post *creacióXat:* es dóna d'alta el xat amb les dades indicades als paràmetres.

context SvXat::getXatsCreatsPerUsuari (usuariCreador:String): Set(TupletType(nomXat:String, tipusXat:String, participants: Set(String), data Creació: date))

post *result* = dades dels xats creats per *l'usuariCreador*.

- **Servei Usuaris** (SvUsu): Aquest servei proporciona les següents operacions:

context SvUsu::getUsuari(usuari: String): TupleType(usuari:String, nom: String, email: String, contactes: Set(us: String))

exc *usuariNoExisteix:* l'usuari indicat al paràmetre no existeix.

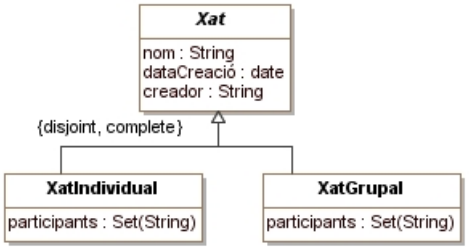
post *result* = telèfon, nom, email i telèfon dels contactes de l'usuari.

context SvUsu::getUsuaris(usuaris: Set(String)): Set(TupleType(usuari:String, nom: String, email: String, contactes: Set(us: String)))

exc *algunUsuariNoExisteix:* algun usuari del conjunt *usuaris* no existeix.

post *result* = telèfon, nom, email i telèfon dels contactes de tots els usuaris.

Com podeu observar tota la informació del nostre sistema està repartida en els serveis anteriors. No obstant això, decidim replicar en el nostre sistema local la informació corresponent als xats (sense els missatges) per dos motius: 1) tenir una còpia dels xats de forma local i 2) el servei de xats té una disponibilitat molt baixa i volem poder crear xats consistents encara que el servei no estigui disponible. Noteu que la baixa disponibilitat del servei de xats pot provocar que, en un moment determinat, els xats del sistema local encara no estiguin incorporats al servei. A continuació disposeu del diagrama de classes del nostre sistema local:



Es demana el diagrama de seqüència de l'operació *AltaXatGrupal*. **Com a criteris de disseny volem minimitzar el nombre d'invocacions remotes mantenint els criteris habituals de reusabilitat i canviabilitat.** Es vol que l'operació dissenyada tingui operacions que puguin ser reusades quan es dissenyi l'operació *AltaXatIndividual*.

2. [0,5 punts] Explica **clarament** si l'excepció *mínimParticipantsNoOk* podria passar a ser la precondició *mínimParticipantsOk* i com la pantalla associada al cas d'ús la podria garantir.

3. [4,5 punts] Considerant el diagrama de classes i restriccions d'integritat donats a l'exercici 1.

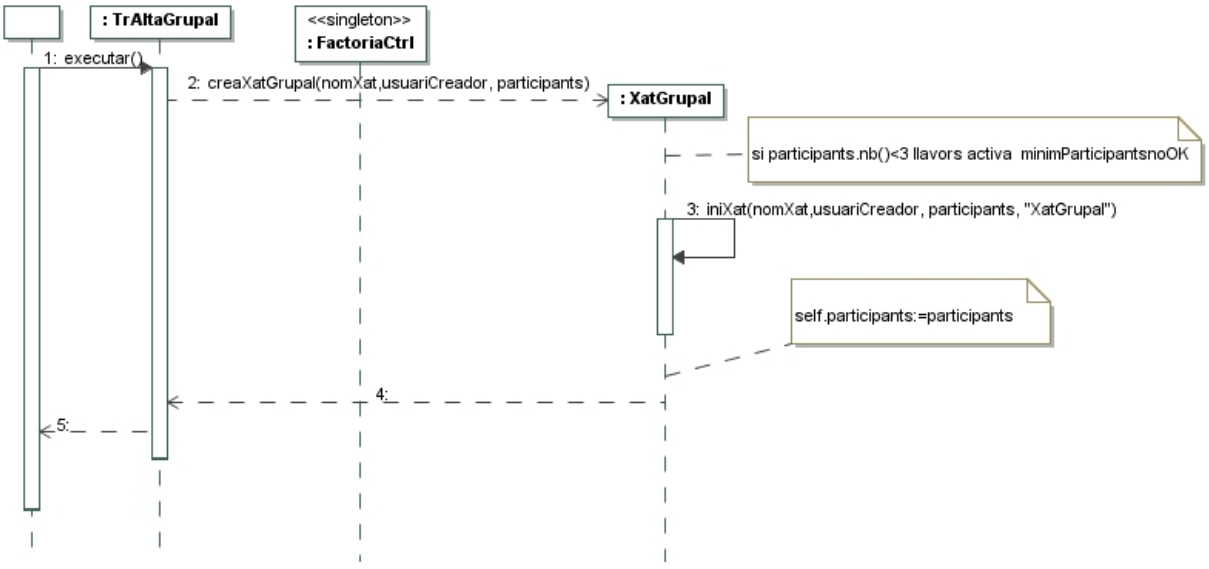
a) [2 punts] Disseny de la base de dades usant Class Table Inheritance per a la jerarquia de xats. Useu restriccions de clau i altres restriccions de columna i de taula sempre que es pugui **(No es poden usar triggers**).

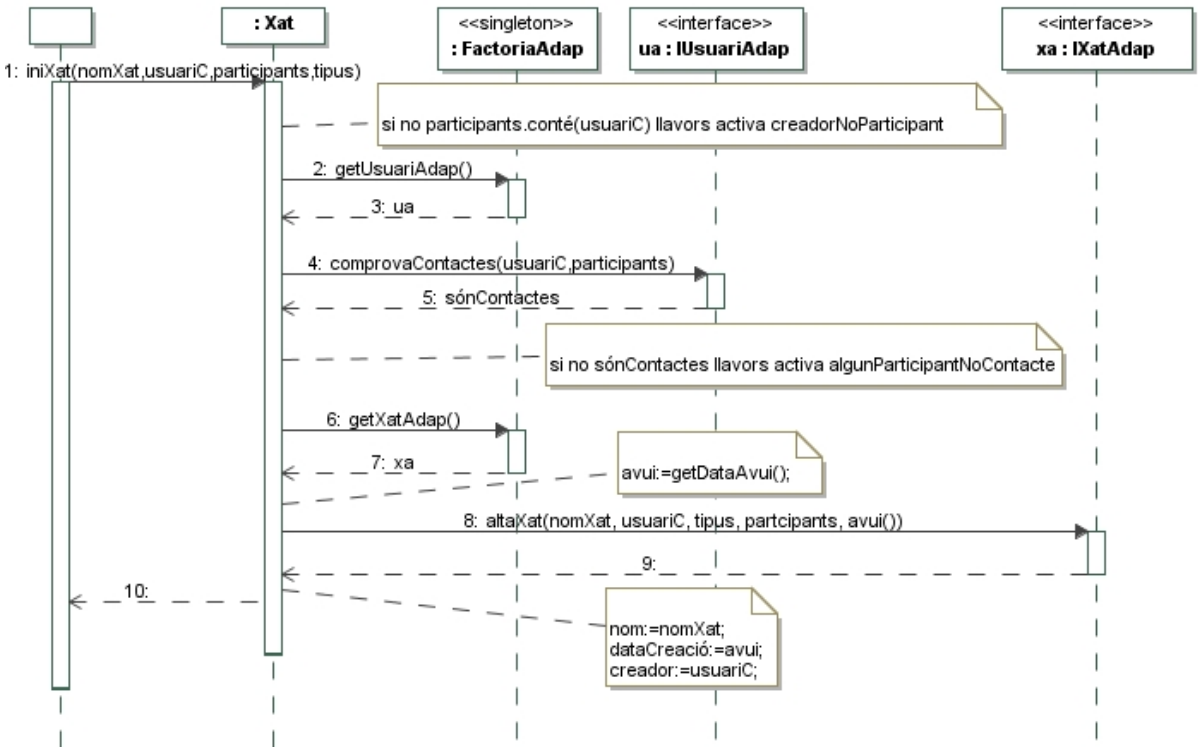
b) [2.5 punts] Disseny de l'operació *AltaXatGrupal* de l'exercici 1 considerant que s'afegeix una nova postcondició (a continuació disposeu del nou contracte). Cal usar **el patró Transaction Script amb Row Data Gateway** (Pasarel.la Fila) (**sense operacions arbitràries de consulta**). No es poden usar els *SvUsu* i *SvXat*. Mostreu clarament en quin moment es llancen les excepcions indicades en l'operació.

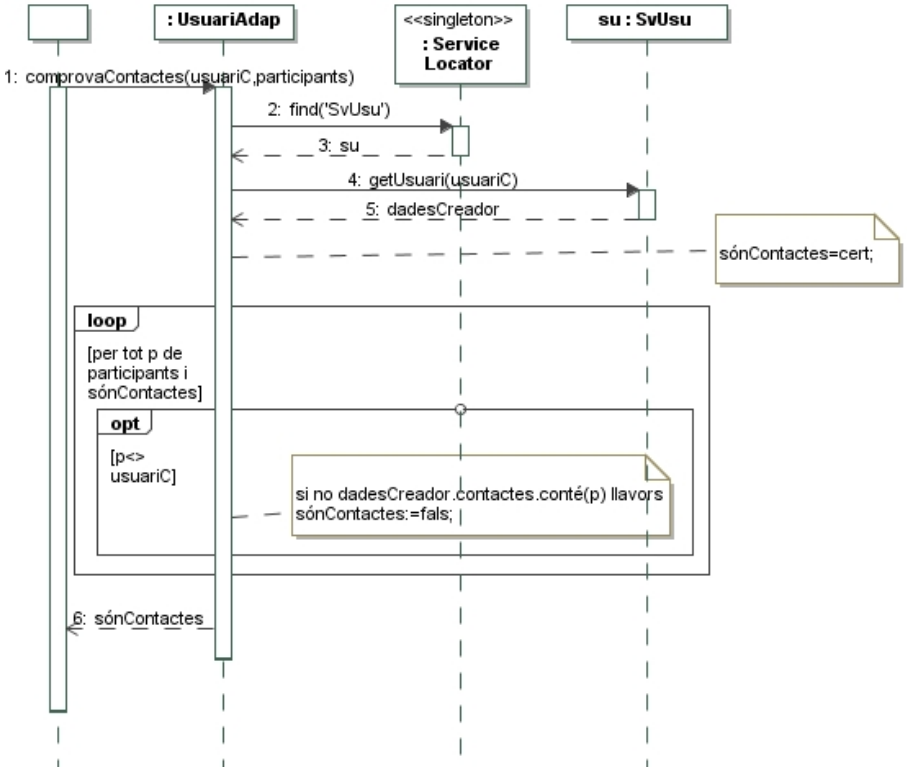
context CapaDomini:: *AltaXatGrupal* (nomXat: String, usuariCreador: String, participants: Set(String)): Integer
exc *xatGrupalExisteix:* el xat identificat als paràmetres existeix.
exc *mínimParticipantsNoOk:*.el nombre de participants del conjunt *participants* és menor que 3.
exc *creadorNoParticipant:* l'*usuariCreador* no hi és al conjunt *participants*.
exc *algunParticipantNoContacte:* algun dels participants, sense incloure l'*usuariCreador*, no és un contacte de l'*usuariCreador* .
post *altaXatGrupal*: es dóna d'alta el xat grupal amb data de creació la data d'avui (podeu utilitzar l'operació del sistema *getDataAvui(): date*) i s'inicialitzen les seves dades.
post *resultat* = nombre de xats grupals creats fins al moment per l'usuari creador indicat al paràmetre que tenen més de 5 participants

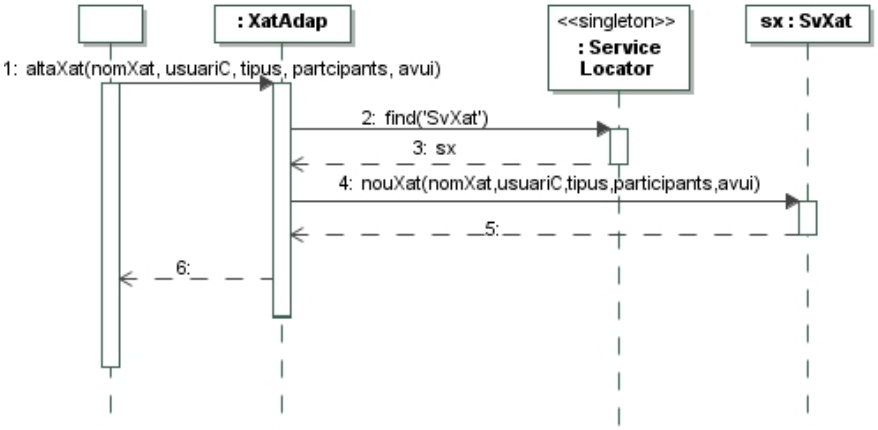
Solució Control QP17-18

1.




: Xat
<<singleton>>
: FactoriaAdap
<<interface>>
ua : IUsuariAdap
<<interface>>
xa : IXatAdap
1: iniXat(nomXat,usuariC,participants,tipus)
si no participants.conté(usuariC) llavors activa creadorNoParticipant
2: getUsuariAdap()
3: ua
4: comprovaContactes(usuariC,participants)
5: sónContactes
si no sónContactes llavors activa algunParticipantNoContacte
6: getXatAdap()
7: xa
avui:=getDataAvui();
8: altaXat(nomXat, usuariC, tipus, partcipants, avui())
9:
10:
nom:=nomXat;
dataCreació:=avui;
creador:=usuariC;


: UsuariAdap
<<singleton>>
: Service
Locator
su : SvUsu
1: comprovaContactes(usuariC,participants)
2: find('SvUsu')
3: su
4: getUsuari(usuariC)
5: dadesCreador
sónContactes=cert;
loop
[per tot p de
participants i
sónContactes]
opt
[p<>
usuariC]
si no dadesCreador.contactes.conté(p) llavors
sónContactes:=fals;
6: sónContactes



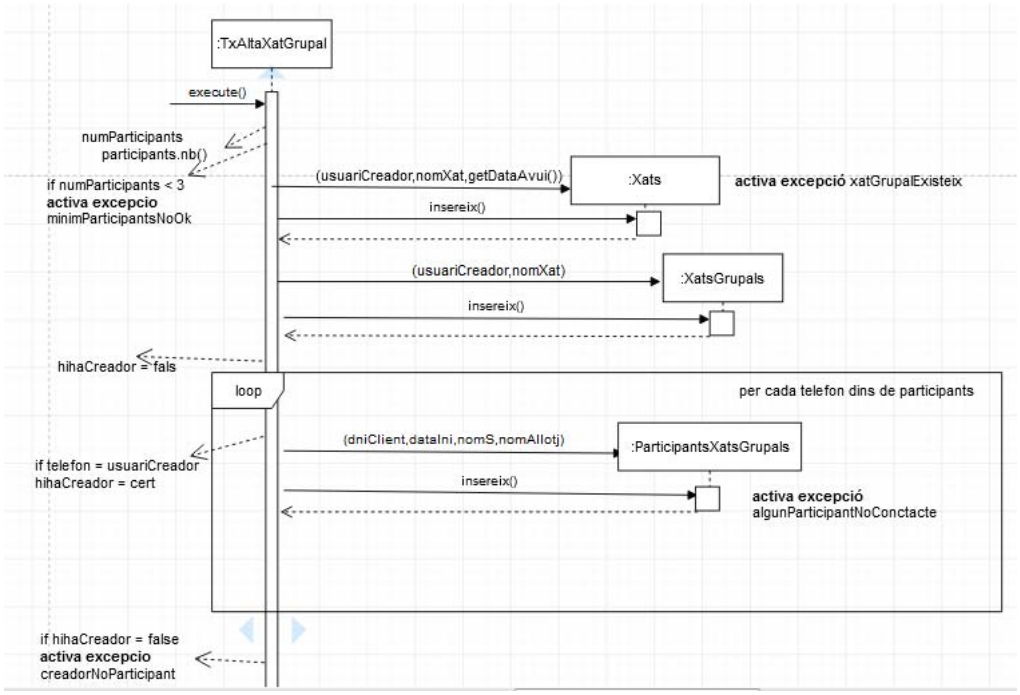
2. La Capa de Presentació pot garantir la precondició si quan anem introduint els participants el botó està desactivat fins que no hem introduït 3 participants.

3. a)

usuaris(telefon, nom, email)
contactes(telefonUsuari, telefonContacte)
{telefonUsuari} referencia usuaris
{telefonContacte} referencia usuaris
xats(telefonCreador, nom, dataCreacio)
{telefonCreador} referencia usuaris
xatsIndividuals(telefonCreador, nom)
{telefonCreador,nom} referencia xats
xatsGrupals(telefonCreador, nom)
{telefonCreador,nom} referencia xats
participantsXatIndividuals(telefonParticipant, telefonCreador, nom)
{telefonParticipant} referencia usuaris
{telefonCreador,nom} referencia xatsIndividuals
participantsXatsGrupals(telefonParticipant, telefonCreador, nom)
{telefonParticipant} referencia usuaris
{telefonCreador,nom} referencia xatsGrupals
missatges(telefonCreador,nom, data,hora, telefonEmisor, text)
{telefonCreador,nom} referencia xats
{telefonEmisor} referencia Usuaris
{telefonEmisor} NOT NULL

RI1 - PKs
RI2 - No es pot implementar en l'esquema de la base de dades
RI3 - no es pot implementar com a FK perquè no sabem si el xat és individual o de grup, sinó es podria fer com una FK de missatges que referencia a participants
RI4 - No es pot implementar en l'esquema de la base de dades
RI5 - En aquest cas es pot implementar com una FK de participants cap a contactes. Seria afegir tant a participantsXatIndividuals com a participantsXatsGrupals la FK següent:
{telefonCreador, telefonParticipant} referencia contactes

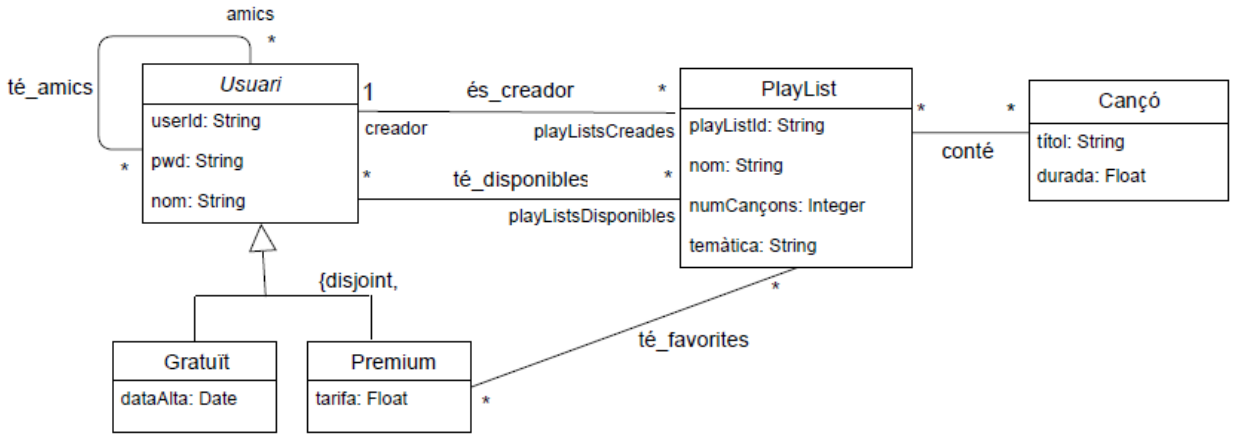
b)


:TxAltaXatGrupal
execute()
numParticipants
participants.nb()
if numParticipants < 3
activa excepcio
minimParticipantsNoOk
(usuariCreador,nomXat,getDataAvui())
:Xats
activa excepció xatGrupalExisteix
insereix()
(usuariCreador,nomXat)
:XatsGrupals
insereix()
hihaCreador = fals
loop
per cada telefon dins de participants
(dniClient,dataIni,nomS,nomAllotj)
:ParticipantsXatsGrupals
if telefon = usuariCreador
hihaCreador = cert
insereix()
activa excepció
algunParticipantNoConctacte
if hihaCreador = false
activa excepcio
creadorNoParticipant

Control QT18-19

Una companyia que s'encarrega de gestionar la música d'un conjunt d'usuaris vol modernitzar un dels seus sistemes software. La companyia s'ha adonat que part de la informació que manté el sistema software (representada en l'esquema conceptual) és també proporcionada per serveis software que proporcionen altres departaments de la companyia.

Esquema conceptual de l'especificació:



R.I. Textuals:

RI1 - Claus: (Usuari, userId); (PlayList, playListId); (Cançó, títol);
RI2 - Un usuari no es pot tenir com amic a sí mateix.
RI3 - Un usuari gratuït no pot tenir disponibles més de 20 playlists.
RI4 - Les playlists creades per un usuari estan entre les playlists que l'usuari té disponibles.
RI5 - Les playlists favorites d'un usuari premium han de ser playlists que l'usuari tingui disponibles.
RI6 - L'atribut numCançons d'una playlist és igual al nombre de cançons que conté la playlist.
RI7 – La durada d'una cançó ha de ser sempre superior a 0.
RIx - Altres restriccions no rellevants pel problema

1. [5 punts] Amb l'objectiu de reduir el manteniment de la informació redundant ens han demanat redissenyar el sistema tenint en compte que disposem de serveis que gestionen la informació dels usuaris i la informació de les playlists dels usuaris. Els serveis dels que disposem són:

- **Servei Usuaris** (SvUsers): gestiona informació dels usuaris del sistema. Proporciona las següents operacions:

```
context SvUsers::newUser(userId:String, pwd:String, name:String)
exc usuariExisteix: l'usuari amb userId ja existeix
post result= registra l'usuari amb userId, pwd i name.

context SvUsers::getInfoUser(userId:String): TupleType (pwd:String,
name:String)
exc usuariNoExisteix: l'usuari amb userId no existeix
post result= retorna el pwd i name de l'usuari amb userId.
```

- **Servei de Gestió de Play Lists** (SvPlayLists)**.** Registra playlists i las seves cançons. Proporciona les següents operacions:

```
context   SvPlayLists::newPlayList(idList:   String,   nameList:String,
temàtica:String titleSongs:Set(String))
exc playListExist= idList ja existeix
```

exc *songNotExist*= alguna cançó del conjunt titleSongs no existeix i el nombre de cançons del conjunt titleSongs és mes gran que 0.

post *registerPlayList*= registra la playList amb idList, nameList, numCançons, temàtica i les cançons del set titleSongs.

context SvPlayLists::getAllPlayLists(): Set(TupleType(idList: String, nameList: String, temàtica:String, numCançons:Int, Set(TupleType (titleSongs:String, duration:Float))))
post *retornaInfoPL*= retorna la informació de totes les playLists (idList, nameList, temàtica, numCançons i el set de títols de les cançons i la seva durada).

context SvPlayLists::getPlayListPerTemàtica(temàtica:String): Set(TupleType (idList:String, nameList:String, temàtica:String, numCançons:Int, Set(TupleType (titleSongs: String, duration:Float))))
post *retornaInfoPLTemàtica*= retorna la informació de totes les playlist de la temàtica donada. Aquesta informació és idList, nameList, temàtica, numCançons, set de títols de les cançons i la seva durada

a) [1 punt] Diagrama de classes del paquet Domain Model de la capa de domini del sistema. Heu d'incloure els atributs (però no les operacions). Heu de considerar que les navegabilitats inicials són: en l'associació *té_disponibles* és de *Usuari* cap a *PlayList (*rol *playListDisponibles)*. En l'associació *té_favorites* es de *Premium* cap a *PlayList*. En la associació *és_creador* es de *PlayList* cap a *Usuari (*rol *creador)*. Recordeu que com a conseqüència del vostre disseny, si és necessari, podeu afegir noves navegabilitats. Com a criteris de disseny volem minimitzar el nombre d'invocacions remotes. També volem mantenir els criteris habituals de reusabilitat, canviabilitat i minimitzant la redundància de les dades.

b) [4 punts] Diagrama de seqüència de l'operació *CreaPremiumIAssociaPLFavoritesPerTemàtica*. Heu de considerar que l'operació que crea usuaris Premium ha de ser altament reusable i que pot ser utilitzada en altres casos d'ús per crear usuaris Premium amb playlists favorites dels seus amics i amb playlists favorites creades pel mateix usuari.

context CapaDomini :: creaPremiumIAssociaPLFavoritesPerTemàtica (userId: String, pwd: String, nom: String, tarifa: Float, temàtica: String)
pre temàticaExisteix: la temàtica indicada al paràmetre existeix.
exc *usuariExisteix*: L'usuari identificat per *userId* ja existeix.
post *creaPremium*: Es crea l'usuari premium amb l'userdId, pwd, nom i tarifa.
post *afegirPLFavoritesPerTemàtica:* S'associen a l'usuari premium creat com a playlists favorites, aquelles playlist que tinguin la temàtica indicada al paràmetre i que hagin estat creades per un usuari gratuït amb més de 10 playlists disponibles.
post *afegirPLDisponibles*: S'associen a l'usuari creat com a playlists disponibles, les playlists favorites de la postcondició anterior.

2. [1 punt] Explica **clarament** com pot garantir la Capa de Presentació o la interfície (pantalles) la precondició del contracte anterior. Necessites afegir alguna operació al servei SvPlayLists per tal de garantir aquesta precondició? Si és així, dóna el contracte d'aquesta operació del servei.

3. [4 punts] Considereu el diagrama de classes i restriccions d'integritat donats (tingueu en compte que en aquest exercici no es tenen disponibles els serveis).

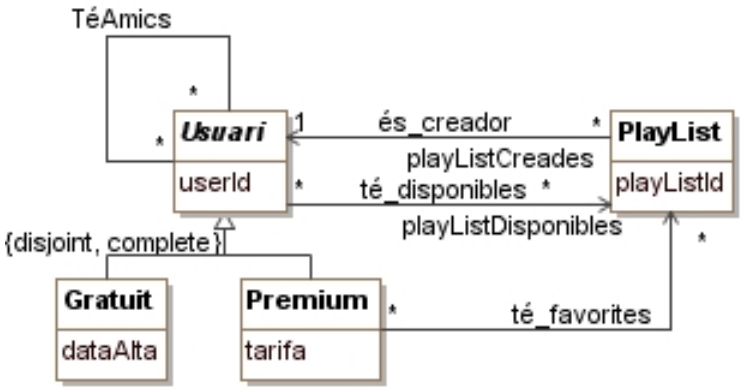
a) **[1.5 punts]** Doneu el disseny de la base de dades usant Class Table Inheritance per a la jerarquia de Usuari. Useu restriccions de clau i altres restriccions de columna i de taula sempre que es pugui **(No es poden usar triggers**).

b) **[2.5 punts]** Dissenyeu l'operació *baixaPlayList* usant **el patró Transaction Script amb Row Data Gateway** (Pasarel.la Fila) (**sense operacions arbitràries de consulta**). Mostreu clarament en quin moment es llancen les excepcions indicades en l'operació.

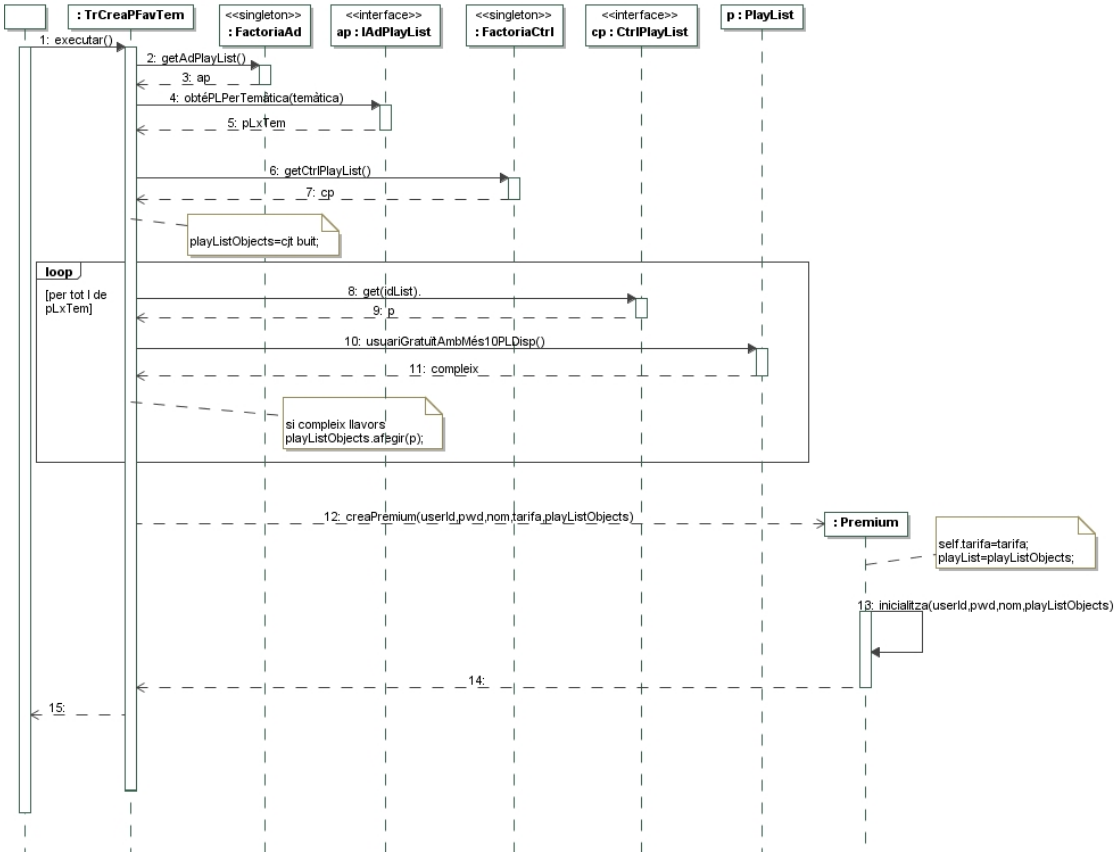
context CapaDomini :: baixaPlayList (usIdCr: String, plId: String): List(nomUsuari)
pre *userExisteix*: L'usuari identificat per usIdCr existeix.
exc *playListNoExisteix*: La playlist identificada per *nomPl* no existeix.
exc *userNoCreador*: L'usuari identificat per usIdCr no és el creador de la playlist nomPl.
exc *ésFavorita*: Hi ha un o més usuaris premium que la té com a favorita.
post *baixaPlayList*: S'elimina la playlist, eliminant també les seves associacions amb les cançons que contenia i amb l'usuari creador.
post *baixaDisponibles*: La playlist deixa d'estar disponible pels usuaris que li tenien.
post *usuarisDisponible*: Es retorna el nom dels usuaris que tenien la playlist com a disponible.

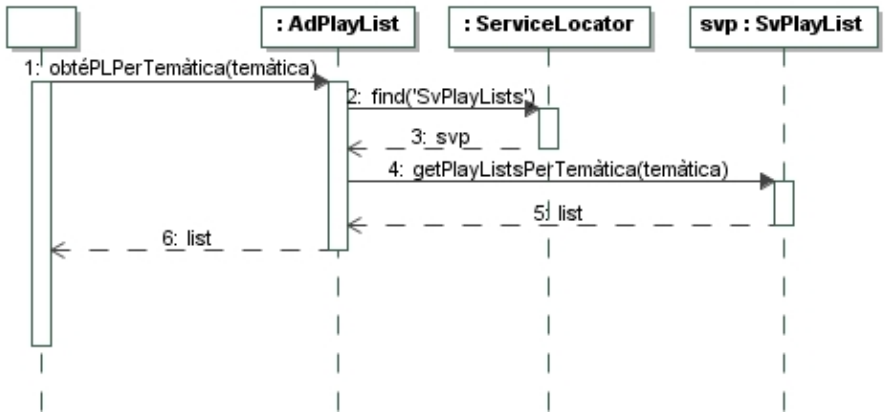
Solució Control QT18-19

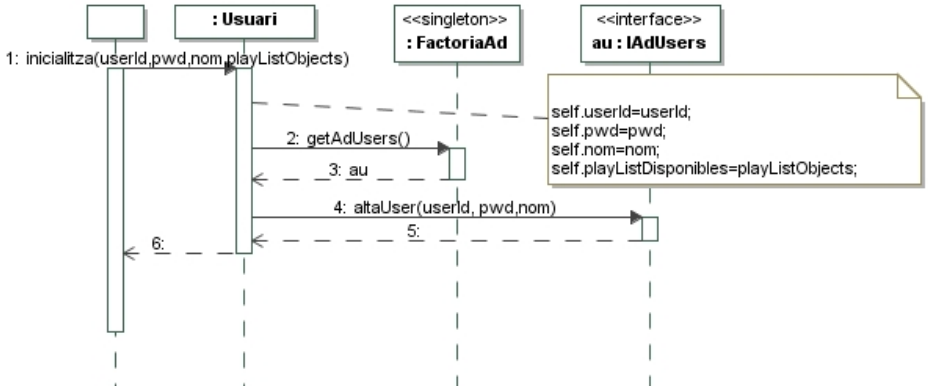
1.- a)

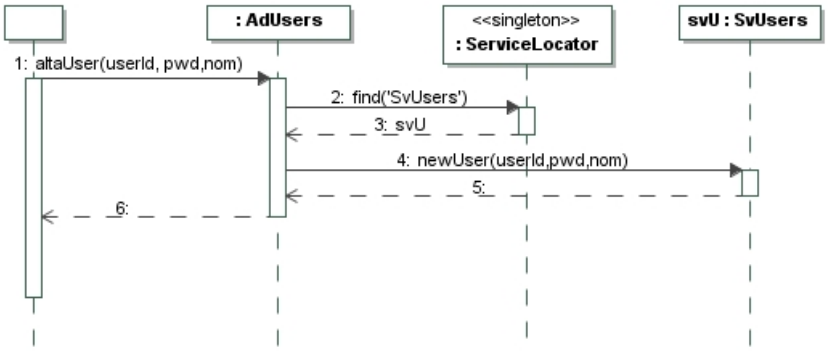


1.- b)


: TrCreaPFavTem
<<singleton>>
: FactoriaAd
<<interface>>
ap : IAdPlayList
<<singleton>>
: FactoriaCtrl
<<interface>>
cp : CtrlPlayList
p : PlayList
1: executar()
2: getAdPlayList()
3: ap
4: obtéPLPerTemàtica(temàtica)
5: pLxTem
6: getCtrlPlayList()
7: cp
playListObjects=cjt buit;
loop
[per tot l de pLxTem]
8: get(idList)
9: p
10: usuariGratuitAmbMés10PLDisp()
11: compleix
si compleix llavors playListObjects.afegir(p);
12: creaPremium(userId,pwd,nom,tarifa,playListObjects)
: Premium
self.tarifa=tarifa;
playList=playListObjects;
13: inicialitza(userId,pwd,nom,playListObjects)
14:
15:


: AdPlayList
: ServiceLocator
svp : SvPlayList
1: obtéPLPerTemàtica(temàtica)
2: find('SvPlayLists')
3: svp
4: getPlayListsPerTemàtica(temàtica)
5: list
6: list


: Usuari
<<singleton>>
: FactoriaAd
<<interface>>
au : IAdUsers
1: inicialitza(userId,pwd,nom,playListObjects)
self.userId=userId;
self.pwd=pwd;
self.nom=nom;
self.playListDisponibles=playListObjects;
2: getAdUsers()
3: au
4: altaUser(userId, pwd,nom)
5:
6:



2.- Hauríen d'afegir una operació al servei que obtingui totes les temàtiques per poder mostra-les i que la temàtica escollida sigui una de les que existeix al servei.

3.-
usuari(<u>userId,</u> pwd, nom)

gratuït(<u>userId</u>,dataAlta)
{userId} referencia usuari

premium(<u>userId</u>,tarifa)
{userId} referencia usuari

playlist(<u>playListId,</u>nom,numCançons,temàtica,userIdCreador)
{userIdCreador} referencia usuari NOT NULL

disponibles(<u>userId,playListId</u>)
{userId} referencia usuari
{playListId} referencia playList

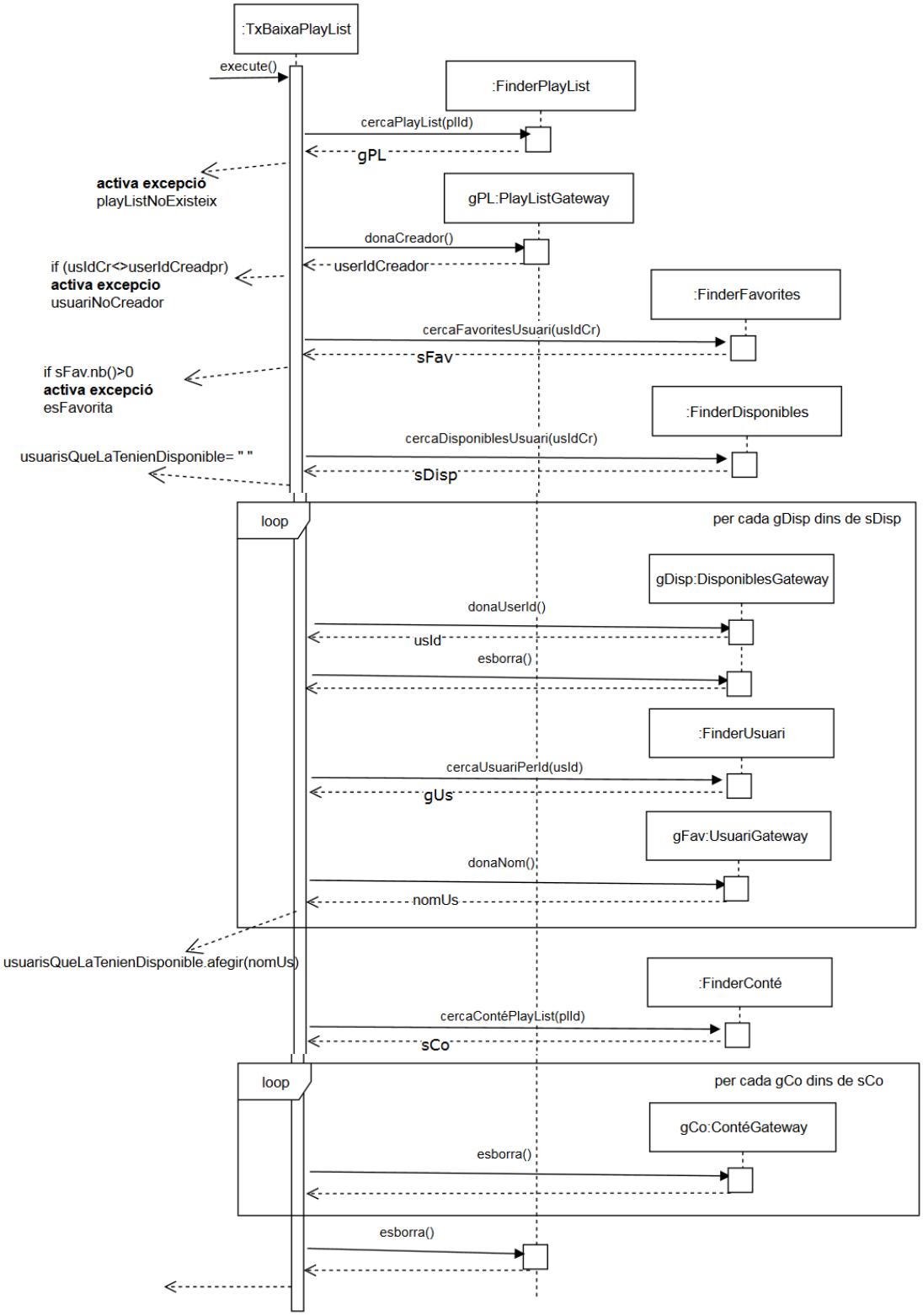
favorites(<u>userId,playListId</u>)
{userId,playListId} referencia disponibles /* RI5 */

cançons(<u>títol</u>, durada)
check(durada>0)

conté(<u>playList, títol</u>)
{playListId} referencia playList
{títol} references cançons

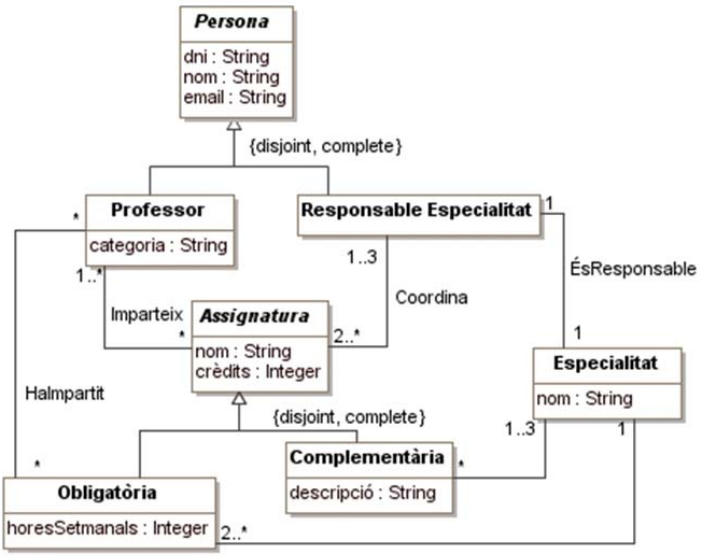
amics(<u>userId, userIdAmic</u>)
{userId} referencia usuari
{userIdAmic} referencia usuari
check (userId <> userIdAmic) /* RI2 */

RI1 - PKs de les taules
RI2 - check de la taula amics
RI3 - no es pot implementar a l'esquema
RI4 - no es pot implementar a l'esquema
RI5 - FK de favorites a disponibles
RI6 - no es pot implementar a l'esquema
RI7 - check a la taula cançons.


:TxBaixaPlayList
execute()
:FinderPlayList
cercaPlayList(plId)
gPL
activa excepció
playListNoExisteix
gPL:PlayListGateway
donaCreador()
userIdCreador
if (usIdCr<>userIdCreadpr)
activa excepcio
usuariNoCreador
:FinderFavorites
cercaFavoritesUsuari(usIdCr)
sFav
if sFav.nb()>0
activa excepció
esFavorita
:FinderDisponibles
cercaDisponiblesUsuari(usIdCr)
usuarisQueLaTenienDisponible= " "
sDisp
loop
per cada gDisp dins de sDisp
gDisp:DisponiblesGateway
donaUserId()
usId
esborra()
:FinderUsuari
cercaUsuariPerId(usId)
gUs
gFav:UsuariGateway
donaNom()
nomUs
usuarisQueLaTenienDisponible.afegir(nomUs)
:FinderConté
cercaContéPlayList(plId)
sCo
loop
per cada gCo dins de sCo
gCo:ContéGateway
esborra()
esborra()

Control QP18-19

1. [5 punts] La FIB vol dissenyar un sistema software per assignar les assignatures a les especialitats, els professors que les imparteixen i els seus responsables. Les assignatures de les especialitats poden ser obligatòries o complementàries (és a dir, optatives de les especialitats). Les obligatòries són d'una única especialitat i les complementàries poden ser de com a màxim 3 especialitats. A continuació disposeu de l'esquema conceptual del sistema a dissenyar:



R.I. Textuals:
RI1 - Claus: (Persona, dni); (Assignatura, nom); (Especialitat, nom);
RI2 - Un responsable només pot coordinar assignatures de l'especialitat de la que és responsable.
RI3 - Les *horesSetmanals* d'una assignatura obligatòria han d'estar entre 2 i 6.
RI4 - Els crèdits de les assignatures complementàries d'una especialitat no poden superar els crèdits de l'assignatura obligatòria de l'especialitat que en té més.
- Altres restriccions no rellevants pel problema

La FIB identifica que hi ha 2 serveis que es podrien utilitzar per mantenir part de la informació del sistema a desenvolupar. La UPC té un servei de directori, anomenat DirectoriUPC, on s'enregistren totes les persones que treballen a tots els centres de la UPC, incloent el centre de la FIB. La FIB, per la seva part, ja té un servei, anomenat AssignaturesFIB, on s'enregistra la informació de totes les assignatures que imparteix, incloent les assignatures de les especialitats. Pels 2 serveis anteriors, us mostrem un subconjunt de les operacions que hi proveeixen amb els contractes corresponents.

context AssignaturesFIB::novaAssignatura (nomA: String, crèditsA: Int, esObligatoria: Bool, hores: Int, desc: String, profesImparteix: Set (dni: String))

exc AssigExisteix: una assignatura amb aquest *nomA* existeix

post es dóna d'alta una assignatura amb tota la informació indicada als paràmetres.

context AssignaturesFIB::obtéAssignatura(nomA: String): TupleType(crèditsA: Int, esObligatoria: Bool, hores: Int, desc: String, profesImparteix: Set (dni: String), profesHanImpartit: Set(dni:String))

exc AssigNoExisteix: l'assignatura identificada per *nomA* no existeix

post retorna la informació de l'assignatura amb *nomA*

context AssignaturesFIB::esborraAssignatura(nomA: String)
exc AssigNoExisteix: l'assignatura identificada per *nomA* no existeix
post s'esborra l'assignatura amb nom *nomA*

context DirectoriUPC::nouProfessor (dni: String, nom: String, cat: String, cent: String, email: String)
exc ProfessorExisteix: el professor amb *dni* existeix
post es dóna d'alta un professor amb DNI *dni*, nom *nom*, centre *cent*, categoria *cat* i correu *email*
context DirectoriUPC::obtéPDI (dni: String): TupleType(nomP: String, catP: String, centreP: String, emailP: String))
exc ProfessorNoExisteix: el professor amb *dni* no existeix
post obté les dades del professor amb aquest *dni*

context DirectoriUPC::obtéProfessorsCentre (centre: String): TupleType(dniP: String, nomP: String, catP: String, emailP: String))
exc centreNoExisteix: el centre no existeix
post obté les dades dels professors del centre indicat al paràmetre

Suposant que la navegabilitat inicial de l'associació *Coordina* va de l'*Assignatura* a *Responsable Especialitat*, la de l'associació entre *Complementària* i *Especialitat* va des de *Complementària* a *Especialitat,* la de l'associació *Imparteix* va d'*Assignatura* a *Professor* i la de l'associació *HaImpartit* va d'*Obligatòria* a *Professor* (recordeu que com a conseqüència del vostre disseny, si és necessari, podeu afegir noves navegabilitats), es demana:

a) **[1 punt]** Diagrama de classes (incloent els atributs però no operacions) del paquet Domain Model de la capa de domini del sistema a desenvolupar. Heu d'incloure tota la informació que no hi sigui en els serveis, tant si és utilitzada per l'operació a dissenyar a l'apartat b) com si no s'utilitza. **Com a criteris de disseny volem minimitzar el nombre d'invocacions remotes mantenint els criteris habituals de reusabilitat, canviabilitat i minimitzant la redundància de les dades**.

b) **[4 punts]** Diagrama de seqüència de l'operació *substituirComplementària* de la capa de domini. Indiqueu clarament en el diagrama de seqüència els singleton, les interfaces i les operacions que són abstractes. A continuació teniu el contracte de la operació.

context CapaDomini::substituirComplementària (nomCompEliminar: String, nomCompAfegir: String, descCompAfegir: String)
exc *compEliminarNoExisteix*: l'assignatura complementària amb nom *nomCompEliminar* no existeix
exc *compAfegirExisteix*: l'assignatura complementària amb nom *nomCompAfegir* existeix
post *baixaAssignatura*: s'elimina l'assignatura amb nom *nomCompEliminar* i les seves associacions
post *creaAssignatura*: es crea una nova assignatura complementària amb el nom *nomCompAfegir i* descripció *descCompAfegir*. La resta de la informació, crèdits,

professors, responsables d'especialitat i especialitats de l'assignatura seran les que tenia l'assignatura eliminada

2. [1 punt] Explica si la Capa de Presentació o la interfície (pantalles) podria garantir que no es llencessin alguna (o les dues) excepcions de la operació anterior (*substituirComplementària*). En cas afirmatiu, expliqueu quina excepció i com es faria. No cal que feu el mapa navegacional.

3. [**4 punts]** Considereu el diagrama de classes i restriccions d'integritat donats a l'exercici 1.

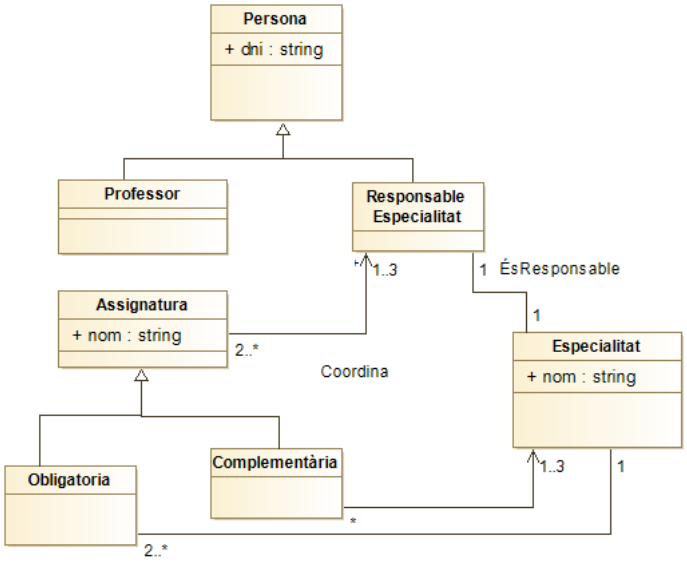
a) [1 punt] Doneu el disseny de la base de dades usant **Concrete Table Inheritance** per a la jerarquia de Persona i **Class Table Inheritance** per a la jerarquia de Assignatura. Useu restriccions de clau i altres restriccions de columna i de taula sempre que es pugui **(No es poden usar triggers**).

b) [3 punts] Dissenyeu l'operació *baixaProfessor* usant **el patró Transaction Script amb Row Data Gateway** (Pasarel.la Fila) (**sense operacions arbitràries de consulta**). Indiqueu clarament en quin moment s'activen les excepcions indicades en l'operació.

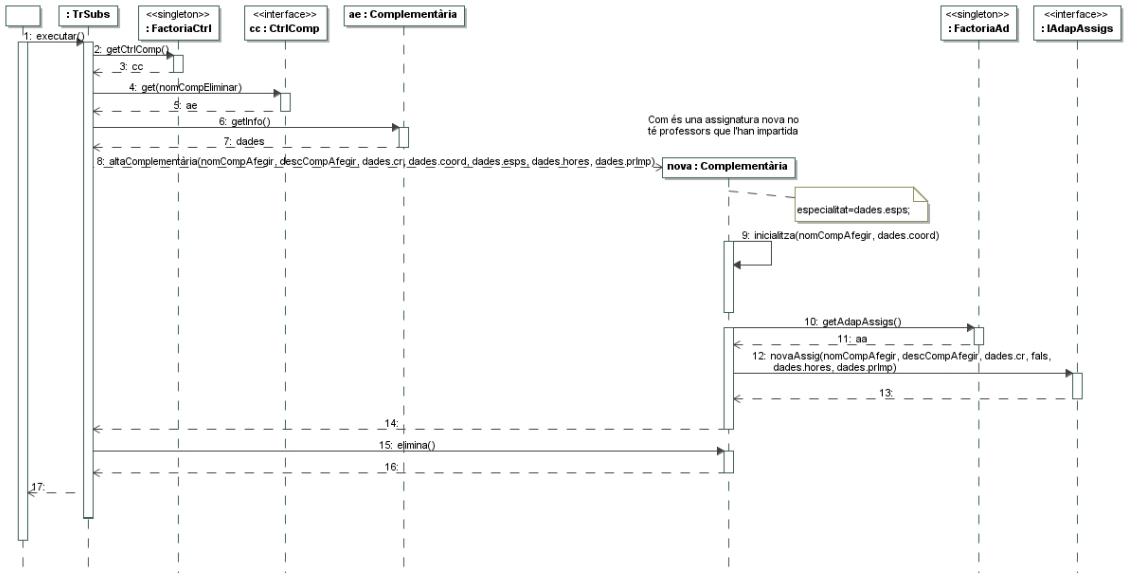
context CapaDomini::baixaProfessor (dniP: String): Set(TupleType(nomObligatoria:String, crèdits: Integer))
exc *professorNoExisteix*: El professor amb dni *dniP* no existeix.
exc *professorImparteixAssignatures*: El professor amb dni *dniP* està actualment impartint assignatures i no es pot eliminar.
post *baixaProfessor*: S'elimina el professor amb dni *dniP*, eliminant també les seves associacions amb les assignatures que ha impartit.
post *donaInformacioObligatories*: Es retorna el nom i número de crèdits de cadascuna de les assignatures obligatòries que el professor amb dni *dniP* ha impartit.

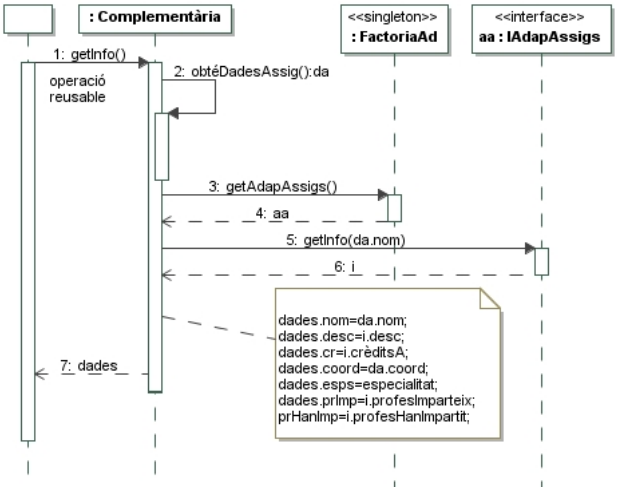
Solució Control QP18-19

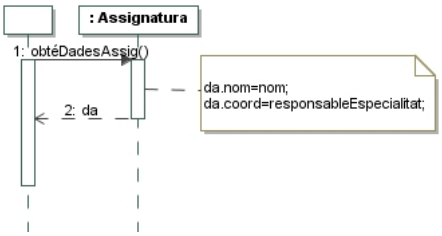
1. a)

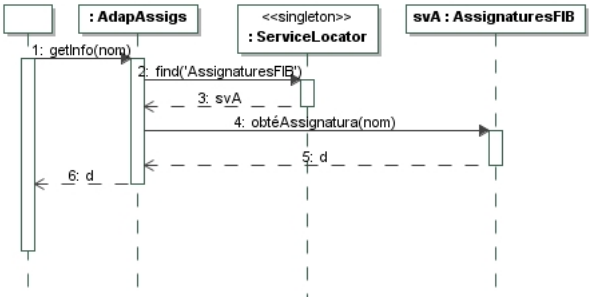


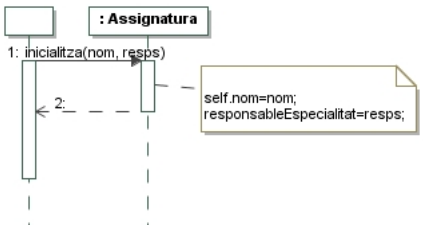
1. b)

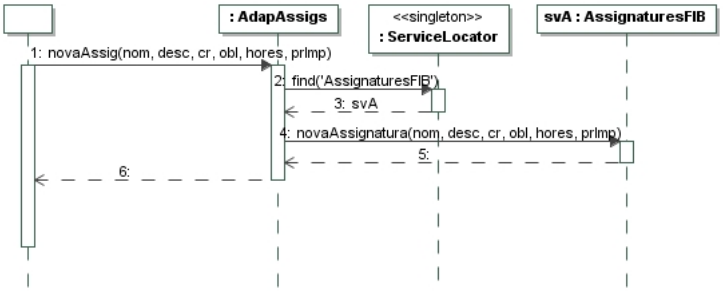


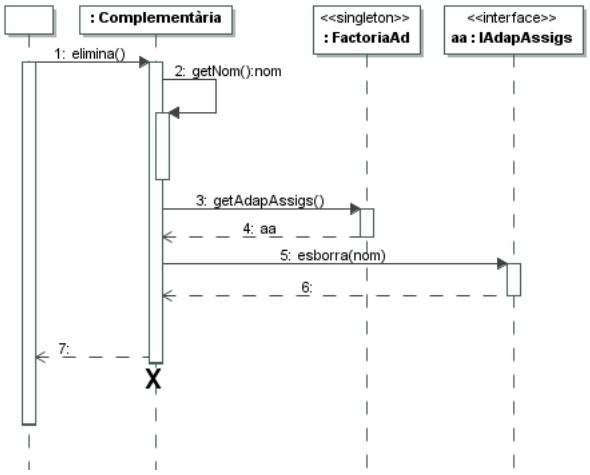

: Complementària
<<singleton>>
: FactoriaAd
<<interface>>
aa : IAdapAssigs
1: getInfo()
operació
reusable
2: obtéDadesAssig():da
3: getAdapAssigs()
4: aa
5: getInfo(da.nom)
6: i
dades.nom=da.nom;
dades.desc=i.desc;
dades.cr=i.crèditsA;
dades.coord=da.coord;
dades.esps=especialitat;
dades.prImp=i.profesImparteix;
prHanImp=i.profesHanImpartit;
7: dades

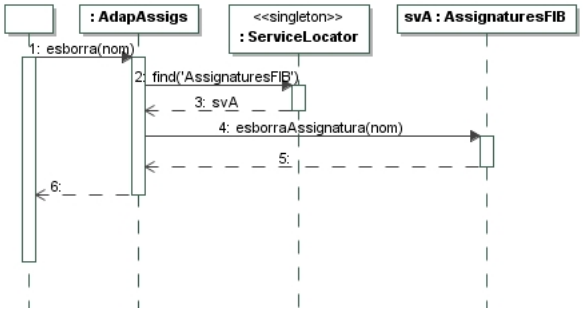

: Assignatura
1: obtéDadesAssig()
da.nom=nom;
da.coord=responsableEspecialitat;
2: da


: AdapAssigs
<<singleton>>
: ServiceLocator
svA : AssignaturesFIB
1: getInfo(nom)
2: find('AssignaturesFIB')
3: svA
4: obtéAssignatura(nom)
5: d
6: d


: Assignatura
1: inicialitza(nom, resps)
self.nom=nom;
responsableEspecialitat=resps;
2:


: AdapAssigs
<<singleton>>
: ServiceLocator
svA : AssignaturesFIB
1: novaAssig(nom, desc, cr, obl, hores, prImp)
2: find('AssignaturesFIB')
3: svA
4: novaAssignatura(nom, desc, cr, obl, hores, prImp)
5:
6:





2. L'excepció *compEliminarNoExisteix* es podria garantir amb la interfície mostrant les complementàries existents en el sistema per tal de que l'usuari seleccioni una complementària que ja existeix. D'aquesta forma l'excepció no s'activaria mai i la condició la complementària a eliminar existeix passa a ser una precondició de l'operació.

3.

especialitats(nomE)

professors(dni, nom, email, categoria)

responsables(dni, nom, email, nomE)
{nomE} referencia especialitats NOT NULL, UNIQUE

assignatures(nomA,credits)

imparteixen(dni, nomA)
{dni} referencia professors
{nomA} referencia assignatures

coordinacions(dni, nomA)
{dni} referencia professors
{nomA} referencia assignatures

obligatories(nomAO,horesSet, nomE)
{nomAO} referencia assignatures

{nomE} referencia especialitats NOT NULL
check(horesSet entre 2 i 6) /*RI3*/

hanImpartit(dni, nomAO)
{dni} referencia professors
{nomAO} referencia obligatories

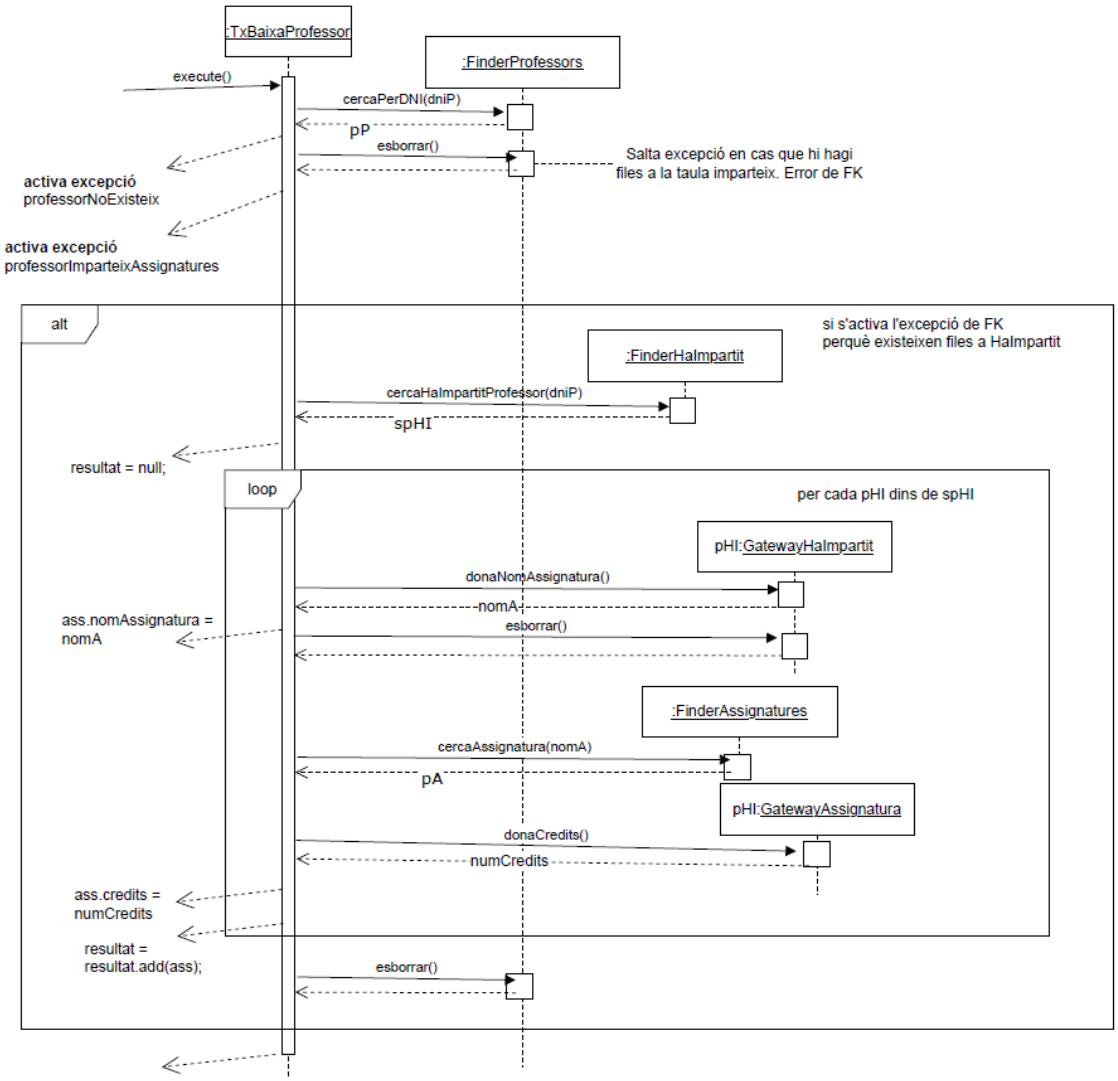
complementaries(nomAC, descripció)
{nomAC} referencia assignatures

tractenDe(nomAC,nomE)
{nomAC} referencia complementaries
{nomE} referencia especialitats

RI1 - PKs de les taules
RI2 - no es pot implementar a l'esquema
RI3 - check taula obligatories
RI4 - no es pot implementar amb restriccions de taula o columna

Noves restriccions per complir amb les múltiplicitats que no es contemplen en l'esquema de la base de dades i que caldria tenir en compte en els dissenys:

- Tota especialitat ha d'estar en exactament 1 fila de la taula responsables.
- Tota assignatura de la taula assignatures ha d'estar com a mínim a 1 fila de la taula imparteixen.
- Tot responsable d'especialitat ha de sortir com a mínim a 2 files de la taula coordinacions
- Tota assignatura ha de sortir com a mínim a 1 fila de la taula coordinacions, i com a màxim a 3 files de la taula coordinacions.
- Tota especialitat ha d'estar en com a mínim 2 files de la taula assignatures obligatories.
- Tota assignatura complementària ha d'estar en com a mínim 1 fila de la taula tractenDe, i com a màxim en 3 files de la taula tractenDe.


:TxBaixaProfessor
:FinderProfessors
execute()
cercaPerDNI(dniP)
pP
esborrar()
Salta excepció en cas que hi hagi files a la taula imparteix. Error de FK
activa excepció
professorNoExisteix
activa excepció
professorImparteixAssignatures
alt
si s'activa l'excepció de FK perquè existeixen files a HaImpartit
:FinderHaImpartit
cercaHaImpartitProfessor(dniP)
spHI
resultat = null;
loop
per cada pHI dins de spHI
pHI:GatewayHaImpartit
donaNomAssignatura()
nomA
ass.nomAssignatura = nomA
esborrar()
:FinderAssignatures
cercaAssignatura(nomA)
pA
pHI:GatewayAssignatura
donaCredits()
numCredits
ass.credits = numCredits
resultat = resultat.add(ass);
esborrar()