MOTORENT

ARNAU SISTACH REINOSO NIUB: 16327555

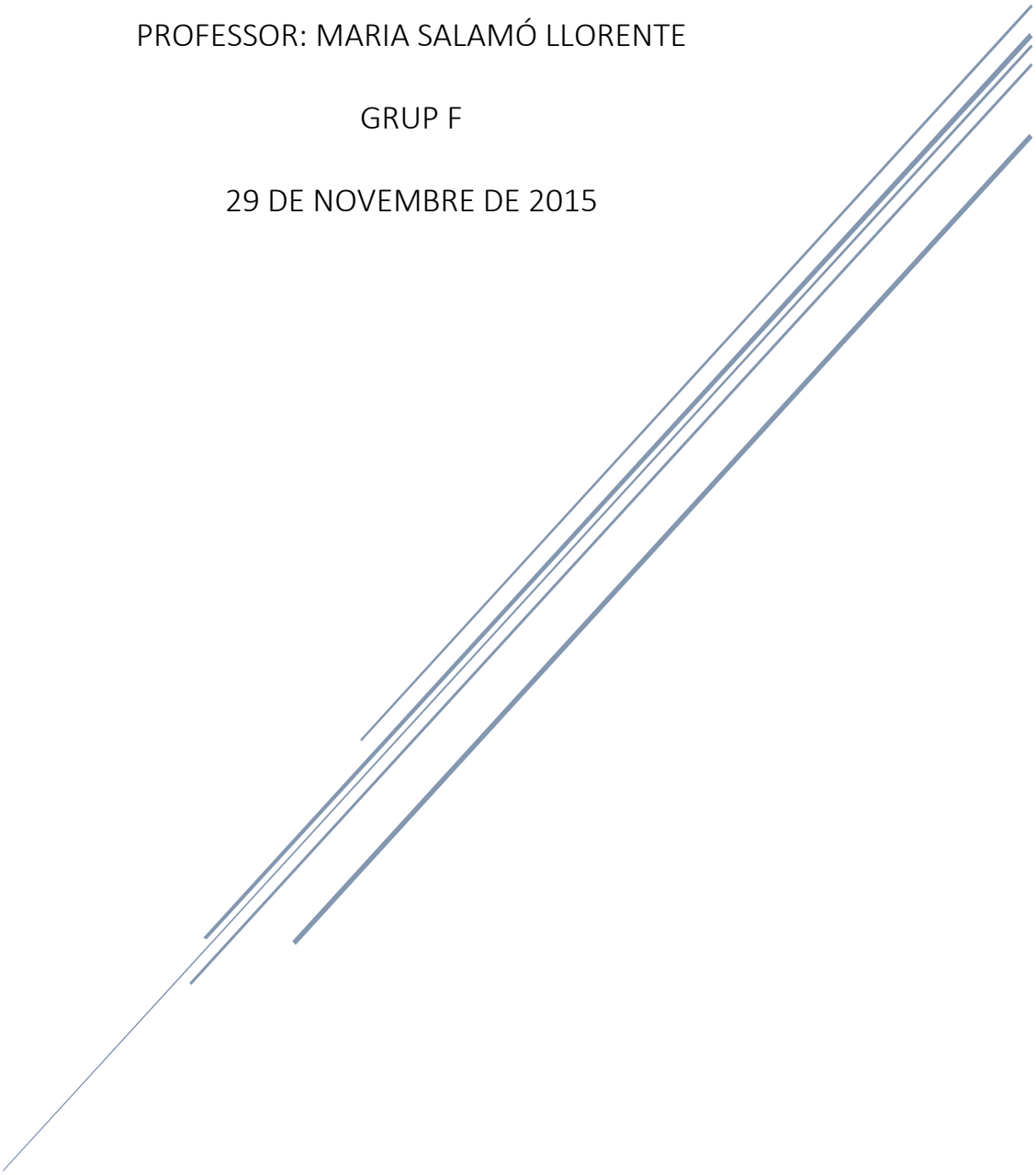
MARC FERRER MARGARIT NIUB: 16624860

ALBERTO LEIVA CABELLO NIUB: 16287434

PROFESSOR: MARIA SALAMÓ LLORENTE

GRUP F

29 DE NOVEMBRE DE 2015



UNIVERSITAT DE BARCELONA
DISSENY DE SOFTWARE

ÍNDEX

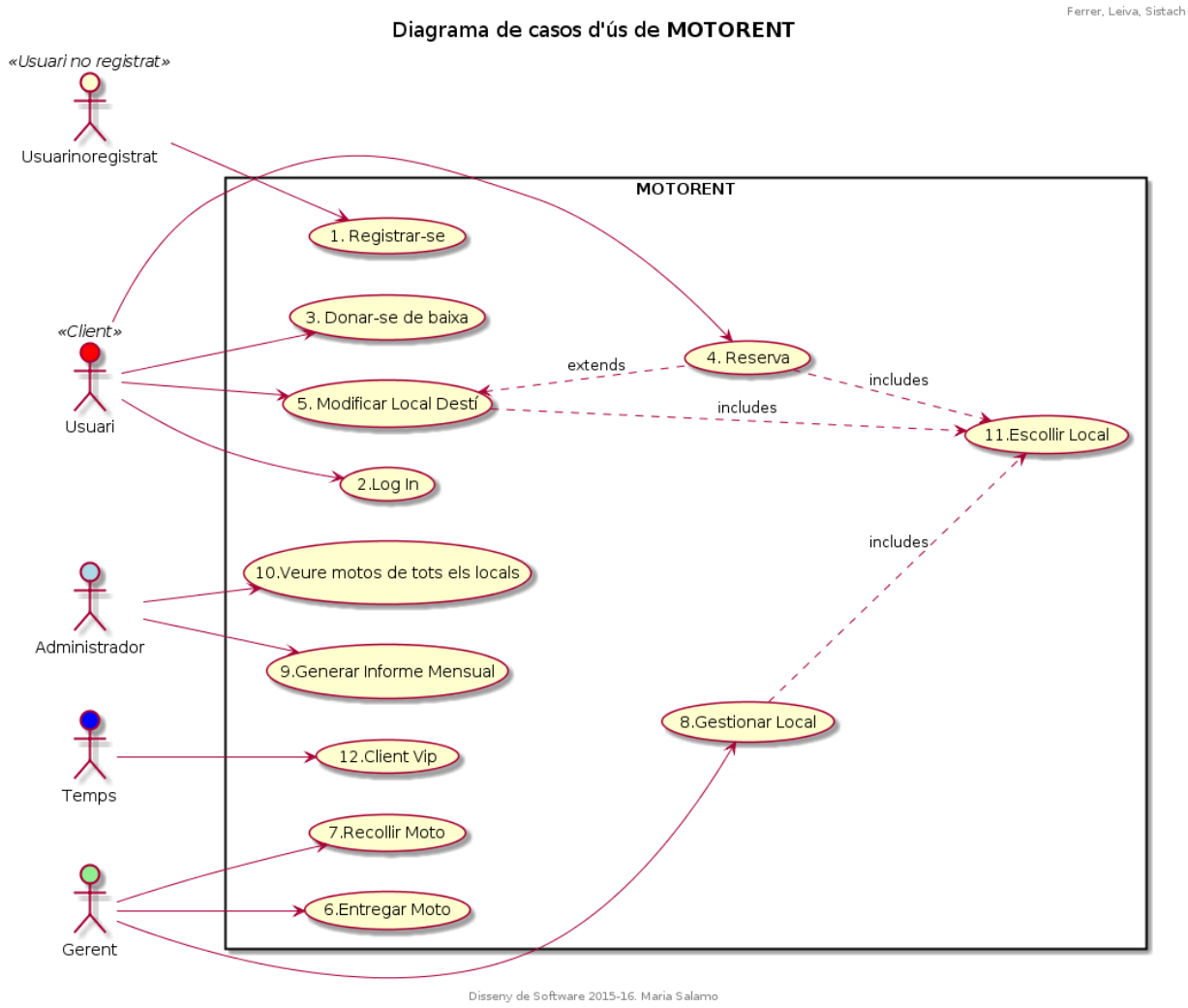
1. INTRODUCCIÓ

L'objectiu d'aquesta pràctica és dissenyar un software per la gestió d'un nou servei de transport públic per desplaçar-se d'un punt a un altre dins de la ciutat de Barcelona. El funcionament d'aquest nou servei es basa en tenir diferents locals repartits en tot Barcelona, els quals cada local disposa d'un cert nombre de motos disponibles per llogar i d'un gerent que és l'encarregat de gestionar el local. D'aquesta manera quan una persona vol llogar una moto, només ha de registrar-se a la web, facilitar les dades necessàries i triar la moto que vol llogar i el temps que la utilitzarà. Un cop l'hagi utilitzat l'usuari retornarà la moto al local corresponent. Així doncs els que ens demanen es dissenyar uns software per poder gestionar aquest servei, és a dir, uns software que permeti a una persona registrar-se i llogar una moto. Aquest software també ha de permetre als gerents gestionar les motos del local.

2. CASOS D'ÚS

Abans de començar a dissenyar aquest software hem de definir una sèrie de requisits, és a dir, hem de definir el comportament del sistema, com usar el sistema per satisfer els objectius que l'empresa ens està demanant i descriure el seu entorn i la relació amb l'entorn. Per poder realitzar això realitzem els casos de ús que ens permetran definir els objectius que s'han de complir. Primer de tot realitzarem un diagrama dels casos d'ús en el qual podrem observar els diferents actors que intervenen en el sistema i les diferents accions que realitzen. Després a partir del diagrama explicarem passa a pas el flux que segueix cada cas d'ús, és a dir, cada acció que realitza un actor quin flux segueix i si hi algun altre flux possible, flux alternatiu.

2.1. DIAGRAMA DE CASOS D'ÚS



Veure codi en PlantUML a l'apartat **Codis**.

Com podem observar hem definit 5 actors. Els actors 'usuari', 'gerent' i 'administrador', són els més bàsics del sistema ja que hem de tenir un usuari, una persona registrada al sistema, que pugui reservar una moto, un gerent que gestioni el local i una persona encarregada de gestionar tot el sistema, de gestionar tota l'empresa. Després també hem definit dos actors més un és la persona que en cara no s'ha registrat, una persona que no s'ha registrat al sistema no pot reservar cap moto, per tant, quan aquesta persona es registri al sistema passarà a ser un usuari. També hem definit un actor temps que és el que s'encarrega de cada cert temps convertir un usuari normal en un usuari VIP.

2.2. CASOS D'US TEXTUALS

Cas d'ús 1

Resum: Un client potencial de Motorent vol registrar-se al sistema de Motorent, per tant ha de crear un usuari i un password, a més d'afegir dades personals. El sistema el registra i ja podrà començar a fer servir els serveis de Motorent.

Actor Principal: Usuari_no_Registat

Precondicions: El client no pot estar ja registrat al sistema.

Flux Bàsic:

1. Usuari_no_Registrat selecciona l'opció de registrar-se.
2. El sistema li demana la informació per poder completar el registre.
    a. Nom d'usuari.
    b. Password.
    c. Nom real.
    d. Cognoms.
    e. DNI.
    f. Telèfon.
    g. Correu electrònic
    h. Data d'inscripció (dia, mes, any).
    i. Adreça on viu (Carrer, número, població, província, codi postal)
    j. Informació del compte corrent (Entitat, oficina, DC, número compte).
3. Usuari_no_Registrat introdueix la informació requerida per el sistema (especificada en el pas anterior).
4. El sistema comprova les dades introduïdes per Usuari_no_Registrat.

5. El sistema dóna d'alta a Usuari_no_Registrat en el sistema amb el seu nom d'usuari i el password introduïts en el pas 3 i el cas d'us finalitza.

Flux alternatiu:

4. a. 1. El sistema detecta dades incorrectes introduïdes per el usuari.

4. a. 2. El sistema mostra un error marcant la informació incorrecte i el flux torna al pas 2.

Cas d'ús 2

Resum: L'usuari ja registrat es vol loguejar al sistema de Motorent fent servir l'usuari i el password fets servir per registrar-se, per poder accedir als serveis que ofereix Motorent.

Actor principal: Usuari.

Precondicions: L'Usuari ha d'estar registrat en el sistema.

Flux Bàsic:

1. L'usuari escull la opció de fer login.
2. El sistema li demana les credencials.
    a. Nom d'usuari.
    b. Password.
3. L'Usuari introdueix les credencials especificades en el pas anterior.
4. El sistema comprova que les credencials introduïdes són correctes.
5. El sistema mostra per pantalla les opcions per als usuaris i el cas d'us finalitza.

Flux Alternatiu:

4. a. 1. Les credencials introduïdes son incorrectes.

4 a. 2. El sistema mostra un error per pantalla dient que les credencials son incorrectes i el flux torna al pas 2.

Cas d'ús 3

Resum: L'usuari ja registrat i logat vol donar-se de baixa en el sistema Motorent per tal de deixar de fer servir els serveis de Motorent

Actor principal: Usuari.

Precondicions: L'usuari ha d'estar logat.

Flux Bàsic:

1. Usuari tria l'opció de donar-se de baixa.
2. El sistema comprova que es pot donar de baixa.
3. El sistema demana que introdueixi el password per poder donar-se de baixa.
4. L'usuari introdueix el password.
5. El sistema de Motorent elimina l'usuari i totes les dades associades introduïdes al cas d'us 1.

Flux Alternatiu:

2. a. 1. L'usuari té una reserva activa.

2. a. 2. El sistema mostra per pantalla un error corresponent a una reserva activa i el cas d'ús finalitza.

4. a. 1. L'usuari introdueix el password erròniament.

4. a. 2. El sistema mostra per pantalla un error corresponen a password incorrecte i el flux torna al pas 3.

Postcondicions: L'usuari es donat de baixa en el sistema de Motorent i ja no podrà fer us dels serveis.

Cas d'ús 4:

Resum: Un client registrat i logat fa una reserva d'una moto al sistema de Motorent, ha de seleccionar la moto, el lloc de recollida i retorn de la moto, així com el temps que la vol i el sistema genera un codi.

Actor Principal: Usuari.

Precondicions: El client està logat.

Flux Principal:

1. L'usuari tria l'opció de Reserva.
2. El sistema comprova les reserves associades al usuari.
3. L'usuari tria el local (Veure cas d'us 11)
4. El sistema mostra els models de motos disponibles (sense desperfectes).
5. Usuari escull el model de la moto.
6. El sistema mostra els colors disponibles per al model de moto que ha escollit Usuari.
7. L'usuari escull el color de la moto.
8. El sistema demana la data per anar a recollir la moto.
9. L'usuari escull la data per anar a recollir la moto.
10. El sistema demana la data per retornar la moto.
11. L'usuari escull la data per retornar la moto.
12. El sistema genera un codi per anar a recollir una moto per al usuari.
13. El sistema registra el codi.
14. El cas d'us finalitza.

Flux alternatiu:

3. a. 1. L'usuari té una reserva activa.

3. a. 2. El sistema mostra un error dient que té una reserva activa i el flux torna al pas 14.

Cas d'ús 5

Resum: L'usuari registrat modifica els local de la reserva.

Actor Principal: Usuari

Precondicions: Usuari ha de tenir una reserva activa (Veure cas d'Us 3).

Flux bàsic:

1. Usuari escull la opció de modificar local destí.
2. El sistema comprova que té una reserva activa.
3. L'usuari tria el local (Veure cas d'us 11).
4. El sistema modifica la reserva de l'usuari on queda reflectit el canvi en els locals.
5. Finalitza el cas d'ús.

Flux alternatiu:

2. a. 1. El sistema detecta que no hi ha una reserva activa.

2. a. 2. El sistema mostra per pantalla un error que diu que no té cap reserva activa.

2. a. 3. El flux torna al pas 5.

Cas d'ús 6

Resum: El client va a local seleccionat a la reserva, entrega el codi i el gerent li entrega la moto.

Actor Principal: Gerent.

Precondicions: El client ha d'haver fet una reserva i el gerent ha d'estar logat.

Flux Bàsic:

1. Gerent escull la opció Entregar moto.
2. El sistema demana que s'introdueixi el codi.
3. Gerent introdueix el codi que li ha donat el client.
4. El sistema comprova que el codi es correcte.
5. El sistema demana si es lliura la moto.
6. Gerent confirma que la moto s'ha lliurat.
7. El sistema registra que la moto està lliurada.
8. El cas d'us finalitza.

Flux alternatiu:

5. a. 1. El codi introduït es incorrecte

5. a. 2. El sistema mostra per pantalla un error per codi incorrecte i el flux torna al pas 8.

Cas d'ús 7

Resum: Arriba X dia, el client arriba al local que ell ha seleccionat com a destí i dona el codi al gerent del local per tal de quedi registrat al sistema, marqui les faltes, i el sistema comprovi que no hi ha retard ni que arribi a les 3 faltes.

Actor principal: Gerent.

Precondicions: Un client té una moto entregada (Veure cas d'us 6) i torna al local amb la moto i el codi.

Flux bàsic:

1. Gerent escull la opció de Recollir moto.
2. El sistema li demana el codi de la reserva.
3. Gerent introdueix el codi.
4. El sistema comprova el temps.
5. El sistema pregunta si té faltes.
6. El gerent introdueix que no hi ha faltes.
7. El sistema afegeix la moto al sistema.
8. El cas d'us finalitza.

Flux alternatiu:

4. a. 1. Gerent introdueix un codi incorrecte.

4. a. 2. El sistema mostra un error corresponent a codi incorrecte i el flux torna al pas 2.

4. a. 1. La reserva te un retard.

4. a. 2. El sistema cobra el retard al client.

4. a. 3. El sistema registra la falta al client.

6. a. 1. El gerent introdueix que si hi ha faltes.

6. a. 2. El sistema pregunta on hi ha les faltes.

6. a. 3. El gerent introdueix on hi ha les faltes.

6. a. 4. El sistema registra les faltes al client.

6. a. 5. El sistema registra els desperfectes de la moto.

Cas d'ús 8

Resum: El gerent sol·licita moure motos d'un local a un altre

Actor principal: Gerent

Precondicions: El gerent esta logat.

Flux bàsic:

1. El gerent selecciona la opció de moure les motos.
2. El gerent tria el local (Veure cas d'us 11).
3. El sistema mostra les motos disponibles per moure.
4. El gerent selecciona les motos que vol moure.
5. El sistema les mou dels locals seleccionats al pas 2.
6. Finalitza el cas d'ús.

Postcondicions: El local del Gerent conté té més de 5 motos.

Cas d'ús 9

Resum: L'Administrador del sistema genera un informe mensual on es mostri per a cada client el total de reserves que ha fet, el local d'origen i destí de cada trajecte, si ha excedit el temps de retorn de la moto, si l'ha retornat en bones o males condicions i el cost total que se li facturarà o se li ha facturat en el seu compte bancari aquell mes.

Actor principal: Administrador

Precondicions: Es final de mes i es vol generar l'informe sobre aquest mes.

Flux bàsic:

1. L'Administrador selecciona l'opció de generar informe.
2. El sistema li demana que entri el mes sobre el qual vol generar l'informe.
3. L'Administrador introdueix el mes del qual vol generar l'informe.

4. El sistema genera i mostra per pantalla l'informe del mes seleccionat.
5. Finalitza el cas d'us.

Flux alternatiu:

5.a.1. El mes introduït no és correcte.

5.a.2. El sistema mostra un error dient que el mes es incorrecte.

5.a.3 El flux torna al pas 4.

Postcondicions: El sistema mostra per pantalla l'informe del mes seleccionat per l'Administrador.

Cas d'ús 10

Resum: Administrador del sistema genera vol veure totes les motos que hi ha a tots els locals.

Actor principal: Administrador

Precondicions: L'administrador ha d'estar logat.

Flux bàsic:

1. L'Administrador selecciona l'opció de veure totes les motos.
2. El sistema li mostra la llista de motos que hi ha a tots els locals.
3. Finalitza el cas d'us.

Cas d'ús 11

Resum: Cas d'us per seleccionar dos locals, el d'inici i final tant per moure motos com per fer una reserva, com per moure motos d'un local a un altre.

Actor principal: Usuari o Gerent

Precondicions: Qualsevol dels actors ha d'estar logat.

Flux bàsic:

1. El sistema mostra els locals disponibles d'inici.
2. L'actor escull el local de sortida.
3. El sistema mostra els locals disponibles on acabar.
4. L'actor escull el local d'arribada

Cas d'ús 12

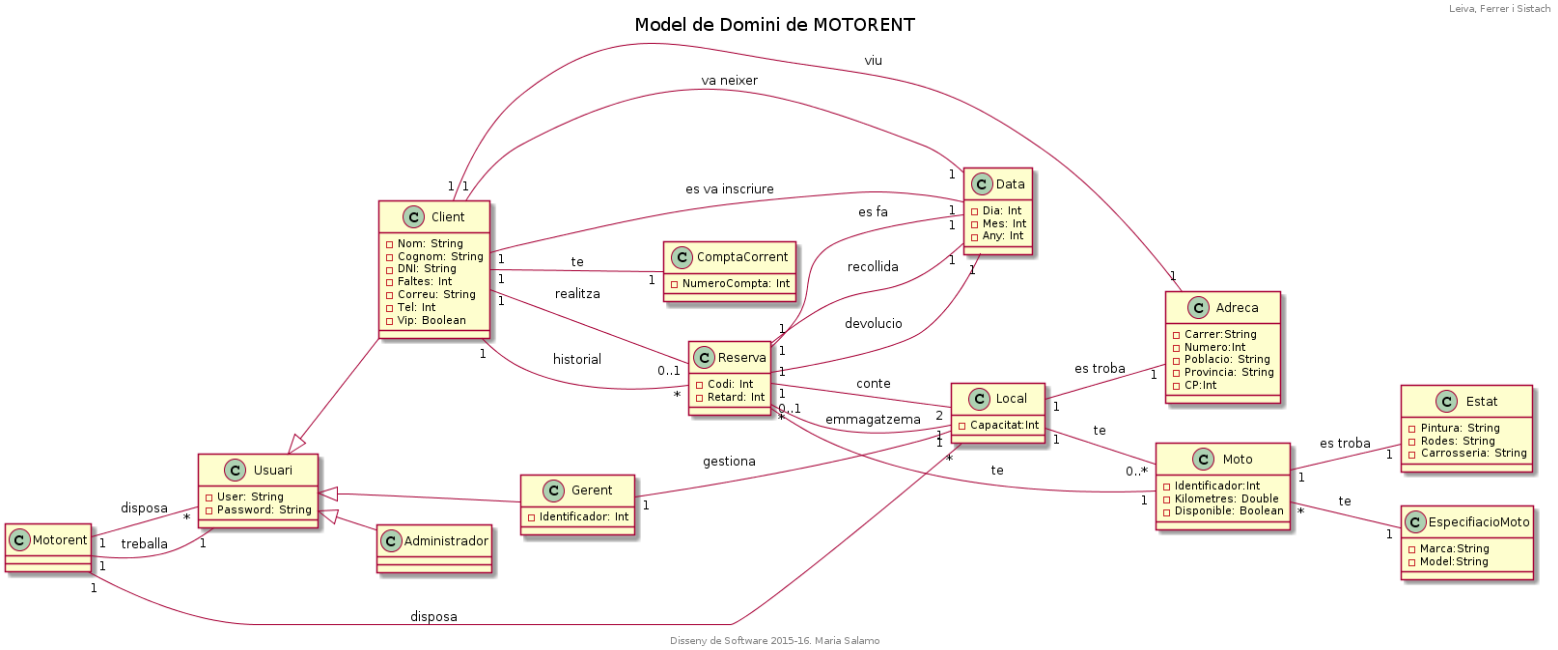
Resum: El sistema assigna clients vip quan passen 2 anys.

Actor principal: Temps.

Flux bàsic:

1. Passen 2 anys per a un usuari registrat.
2. El sistema assigna aquest client l'etiqueta vip
3. El cas d'us finalitza.

3. MODEL DE DOMINI



Veure codi en PlantUML a l'apartat **Codis**.

Com podem observar, els actors creats prèviament en els casos d'ús apareixen en el model. També podem observar que els tres actors client, gerent i administrador, tenen com a classe pare usuari, ja que tots necessiten un nom d'usuari i un password. A part, també apareixen altres classes necessàries com per exemple, la classe Local, Reserva i Moto que són necessàries per poder realitzar les operacions que ens demanen.

A partir d'aquest model de domini podrem realitzar els diagrames de seqüència que són els que en permetran entre que realitza el sistema quan realitzem una acció determinada, és a dir, què fa el sistema quan realitzem un cas d'ús.

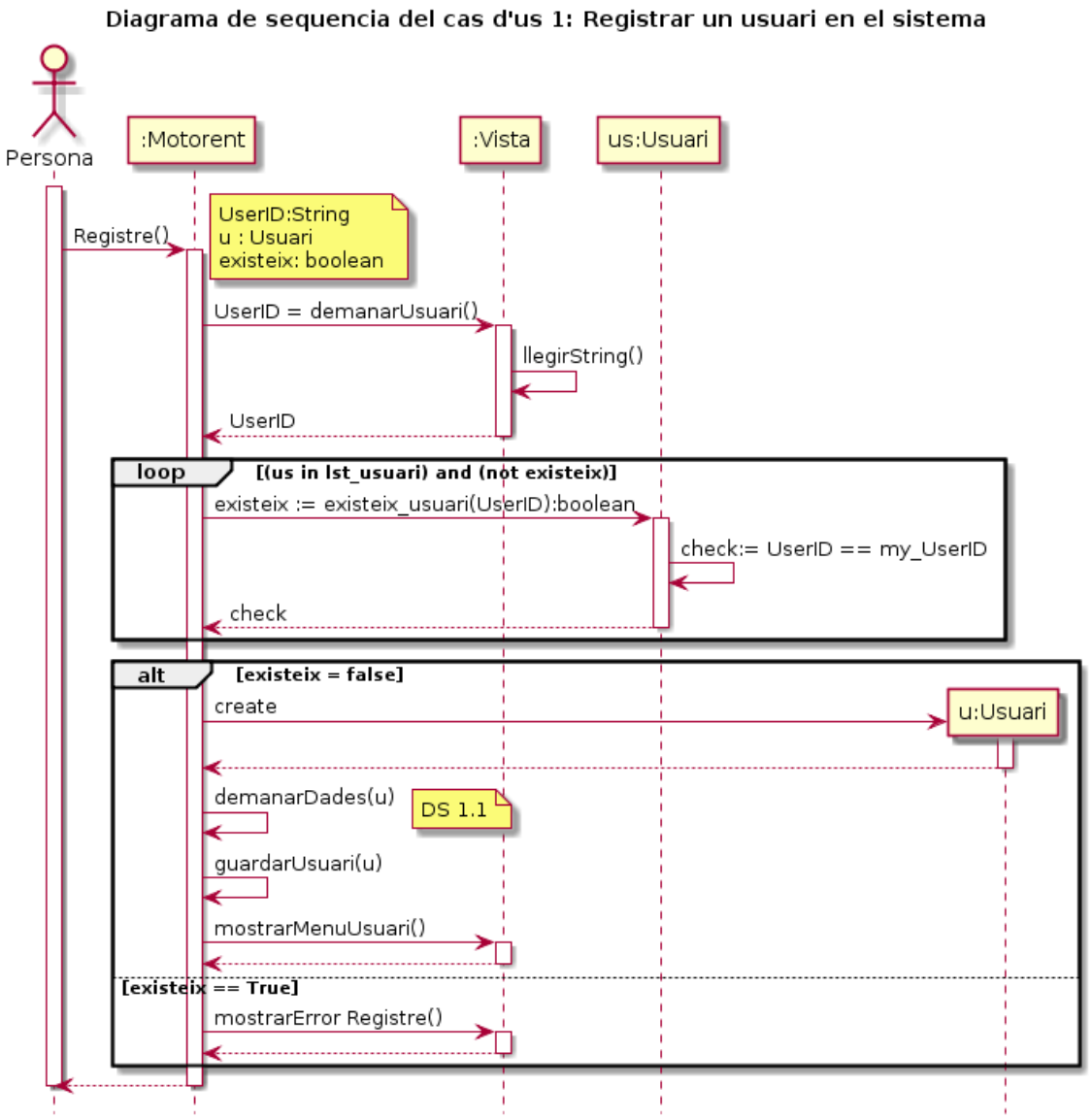
4. DIAGRAMES DE SEQÜÈNCIA

Un diagrama de seqüència mostra la interacció d'un conjunt d'objectes en una aplicació a través del temps i es modela per a cada cas d'ús. El diagrama de seqüència conté detalls d'implementació de l'escenari , incloent els objectes i classes que es fan servir per implementar l'escenari i missatges intercanviats entre els objectes.

A continuació dissenyarem els següents casos d'ús:

1. Registrar usuari en el sistema.
2. Logar usuari en el sistema.
3. Fer la reserva d'una moto.
4. Lliurar la moto reservada a un client.
5. Retornar la moto al local destí.
6. Veure les motos que hi ha en tots els locals.
7. Gestió de motos d'un local (és a dir, moure motos d'un local a un altre quan el gerent ho sol·licita) .
8. Informe al final de cada mes que ara fa l'administrador del sistema.

4.1. REGISTRAR USUARI AL SISTEMA



Veure codi en PlantUML a l'apartat **Codis.**

Aquest diagrama explica el funcionament del sistema quan una persona anònim, una persona que no esta registrada, es vol registrar al sistema. Suposem que el primer que es demana és el seu DNI i es comprova que no hi ha cap usuari amb el mateix DNI, és a dir, que ja està registrat. Un cop es comprova que no està registrat se li demanen les dades i es guarda l'usuari i es mostra el menú d'un usuari registrat.

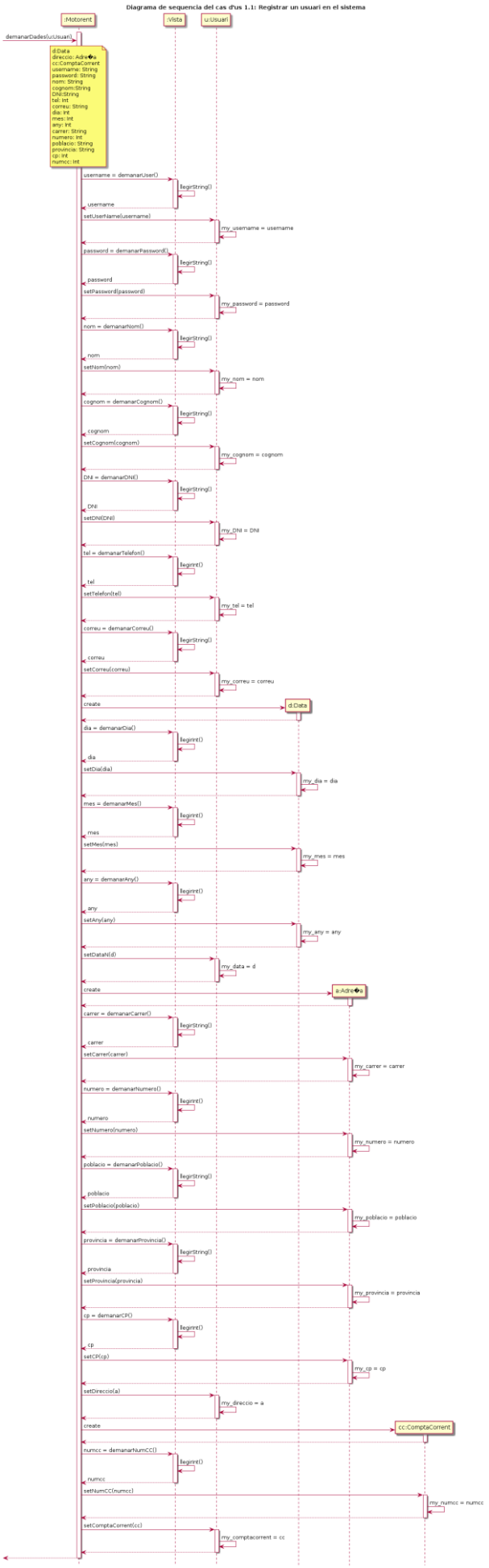
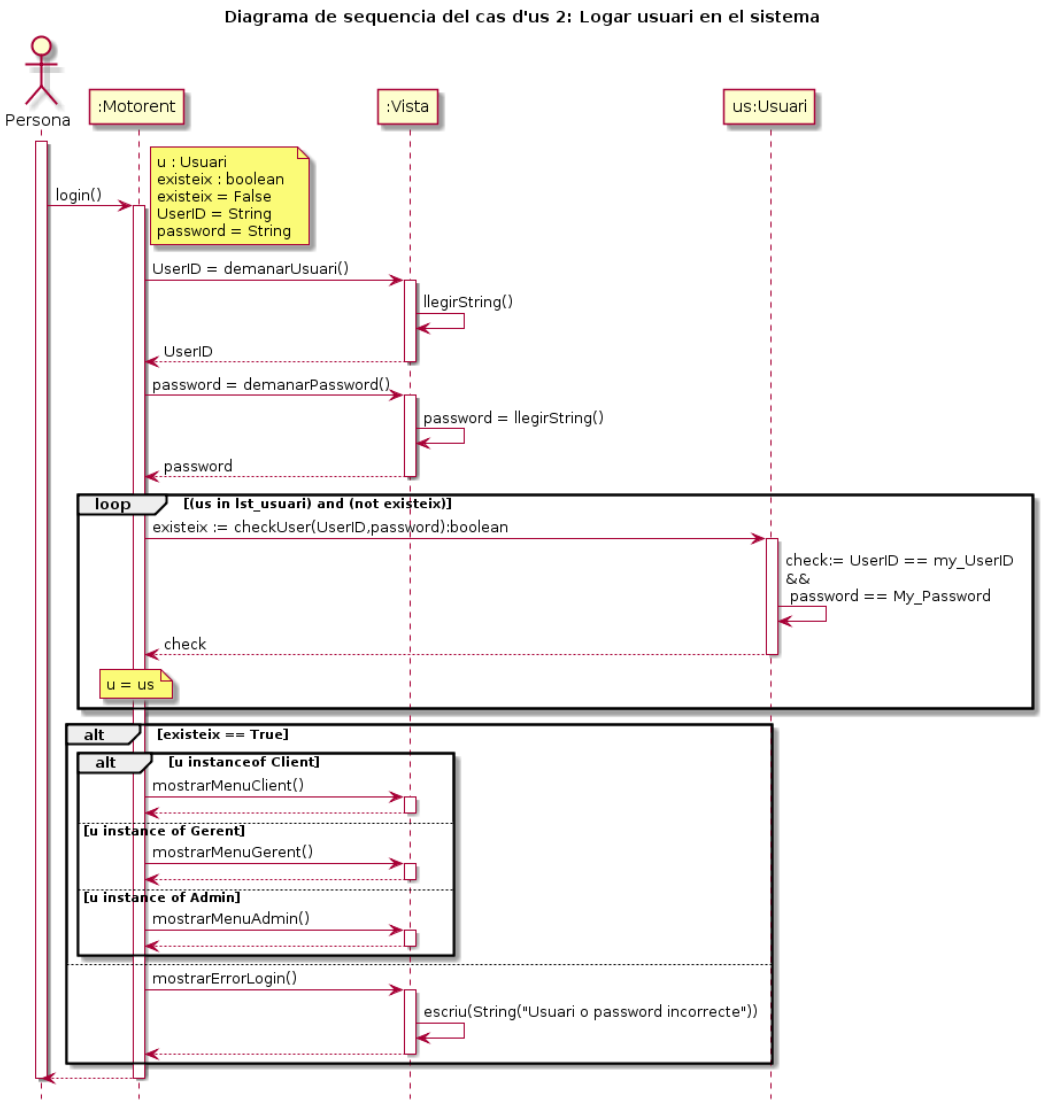


Diagrama de sequencia del cas d'us 1.1: Registrar un usuari en el sistema
:Motorent
:Vista
u:Usuari
demanarDades(u:Usuari)
d:Data
direccio: Adre@a
cc:ComptaCorrent
username: String
password: String
nom: String
cognom:String
DNI:String
tel: Int
correu: String
dia: Int
mes: Int
any: Int
carrer: String
numero: Int
poblacio: String
provincia: String
cp: Int
numcc: Int
username = demanarUser()
llegirString()
username
setUserName(username)
my_username = username
password = demanarPassword()
llegirString()
password
setPassword(password)
my_password = password
nom = demanarNom()
llegirString()
nom
setNom(nom)
my_nom = nom
cognom = demanarCognom()
llegirString()
cognom
setCognom(cognom)
my_cognom = cognom
DNI = demanarDNI()
llegirString()
DNI
setDNI(DNI)
my_DNI = DNI
tel = demanarTelefon()
llegirInt()
tel
setTelefon(tel)
my_tel = tel
correu = demanarCorreu()
llegirString()
correu
setCorreu(correu)
my_correu = correu
create
d:Data
dia = demanarDia()
llegirInt()
dia
setDia(dia)
my_dia = dia
mes = demanarMes()
llegirInt()
mes
setMes(mes)
my_mes = mes
any = demanarAny()
llegirInt()
any
setAny(any)
my_any = any
setData(d)
my_data = d
create
a:Adre@a
carrer = demanarCarrer()
llegirString()
carrer
setCarrer(carrer)
my_carrer = carrer
numero = demanarNumero()
llegirInt()
numero
setNumero(numero)
my_numero = numero
poblacio = demanarPoblacio()
llegirString()
poblacio
setPoblacio(poblacio)
my_poblacio = poblacio
provincia = demanarProvincia()
llegirString()
provincia
setProvincia(provincia)
my_provincia = provincia
cp = demanarCP()
llegirInt()
cp
setCP(cp)
my_cp = cp
setDireccio(a)
my_direccio = a
create
cc:ComptaCorrent
numcc = demanarNumCC()
llegirInt()
numcc
setNumCC(numcc)
my_numcc = numcc
setComptaCorrent(cc)
my_comptacorrent = cc

Veure codi en PlantUML a l'apartat **Codis.**

El diagrama anterior mostra el diagrama de seqüencia 1.1 que és el que correspon a demanar les dades a l'usuari necessàries a l'usuari. En el nostra cas treballem amb el model-vista-controlador per tant la vista demana les dades a l'usuari i aquesta les torna al controlador on seran utilitzades. Com podem observar es creen dos 2 classes, a part de la classe u:Usuari , la classe Data que ens serveix per guardar una data, en aquest cas la data de naixement, i la classe ComptaCorrent que és la compta corrent de l'usuari.

4.2. LOGAR USUARI AL SISTEMA



Veure codi en PlantUML a l'apartat **Codis.**

Aquest diagrama és el de logar-se en el sistema. Primer de tot el que fem és comprovar que ha introduït el nom d'usuari i la contrasenya bé, és a dir, que existeix l'usuari i que ha entrat aquests dos camps correctament. Després de comprovar-ho es mostra el menú de l'usuari segons si aquest es tracta d'un client, un gerent o l'administrador, en cas contrari es mostra un text indicant que ha introduït algun camp malament.

4.3. FER LA RESERA D'UNA MOTO

A continuació es mostra el diagrama de seqüència de fer una reserva d'una moto, primer es demana que es selecciona un local origen (DS 3.1) i un local destí. Un cop s'ha seleccionat el local origen i el destí es demana que seleccioni una moto de les que es troben en el local origen (DS 3.2), aquesta moto ha d'estar disponible i un cop es seleccioni ha d'estar no disponible fins que es retorni la moto. Un cop s'ha seleccionat la moto i els dos locals s'agafa del sistema la data en què s'ha realitzat la reserva i es demana al client la data de recollida de la moto al local i la data de devolució, un cop es tenen no totes les dades necessàries es crea la reserva i es genera un codi que és el que es retorna al client que l'utilitzarà per recollir i retornar la moto.

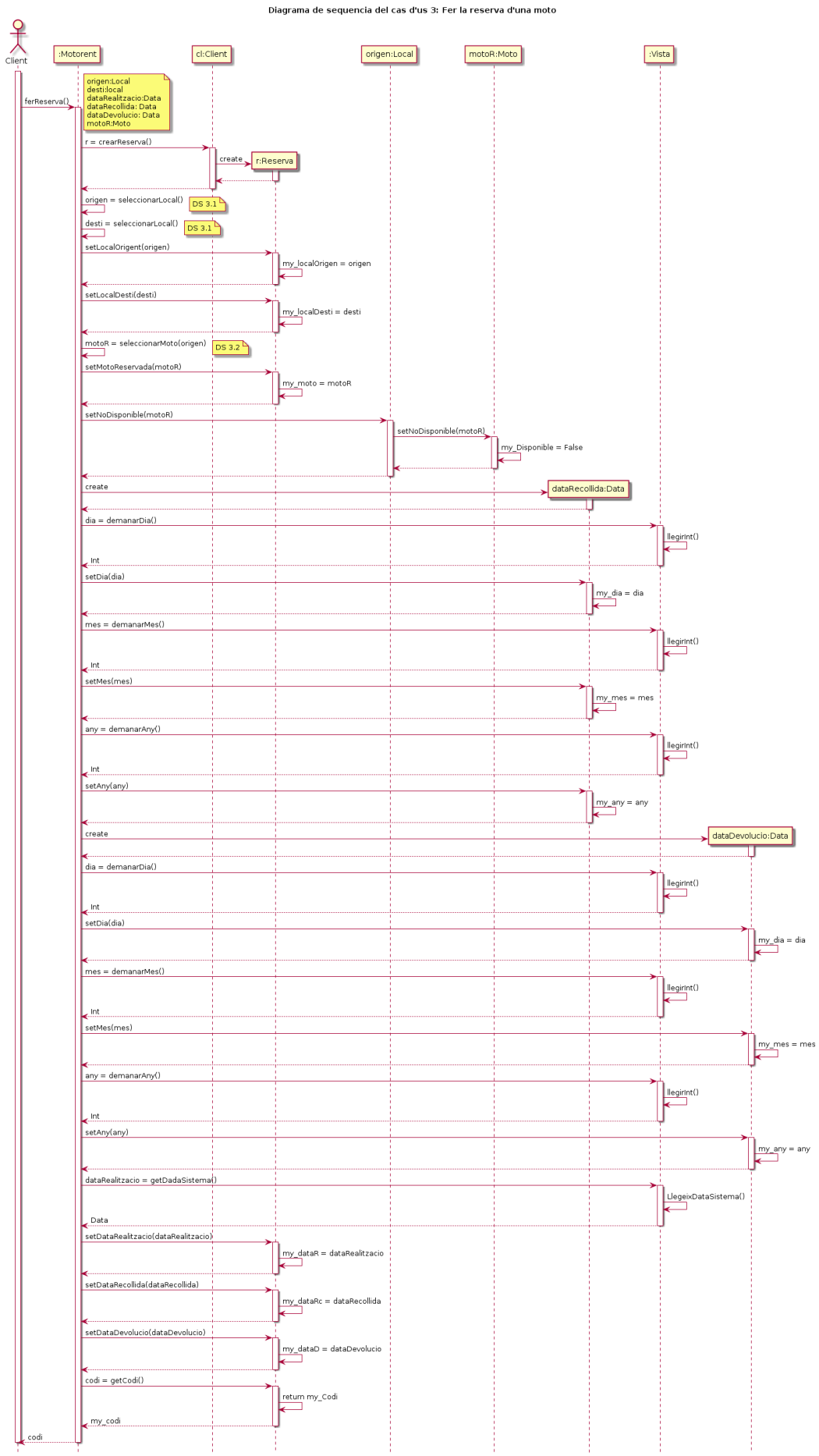
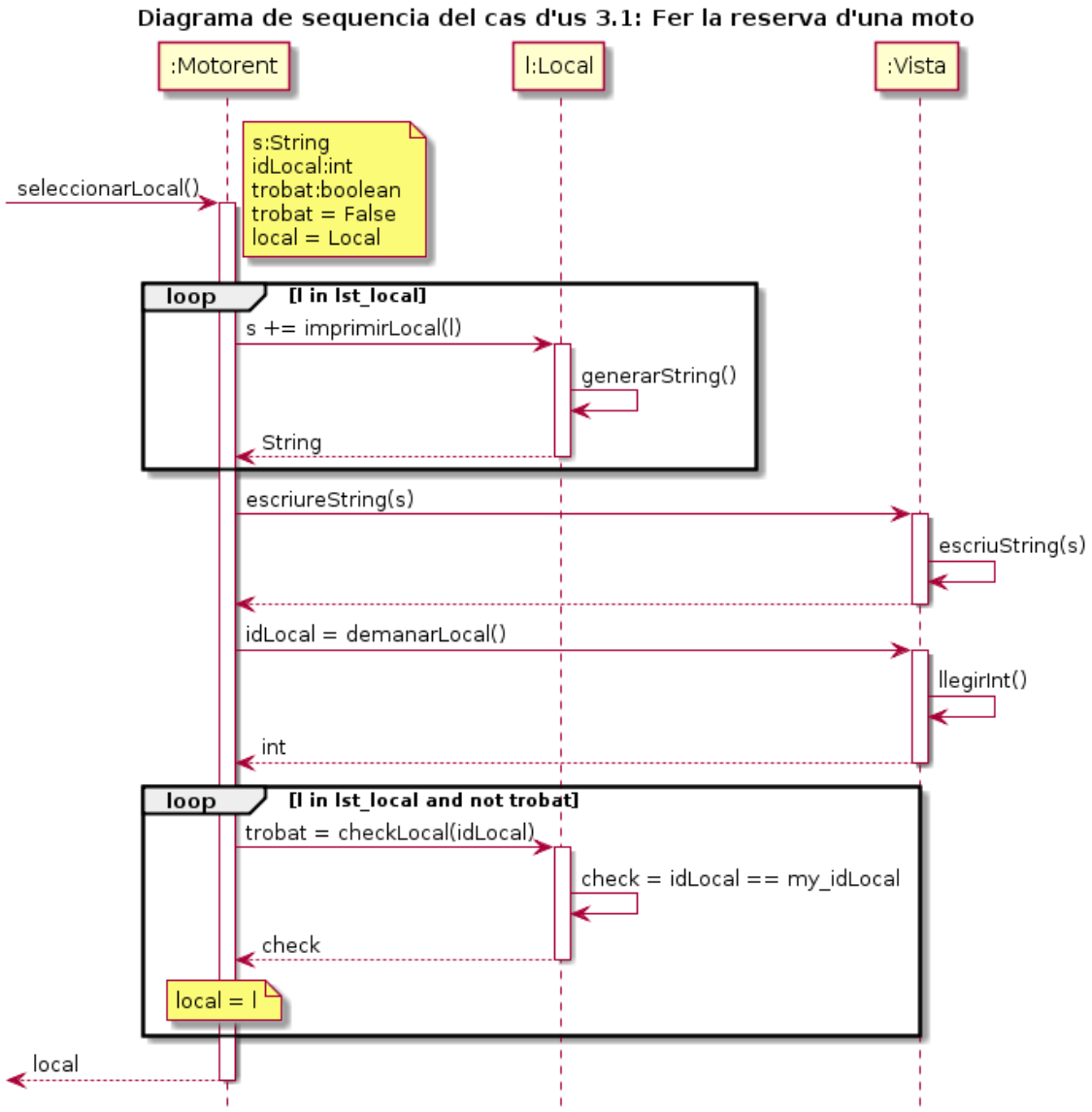
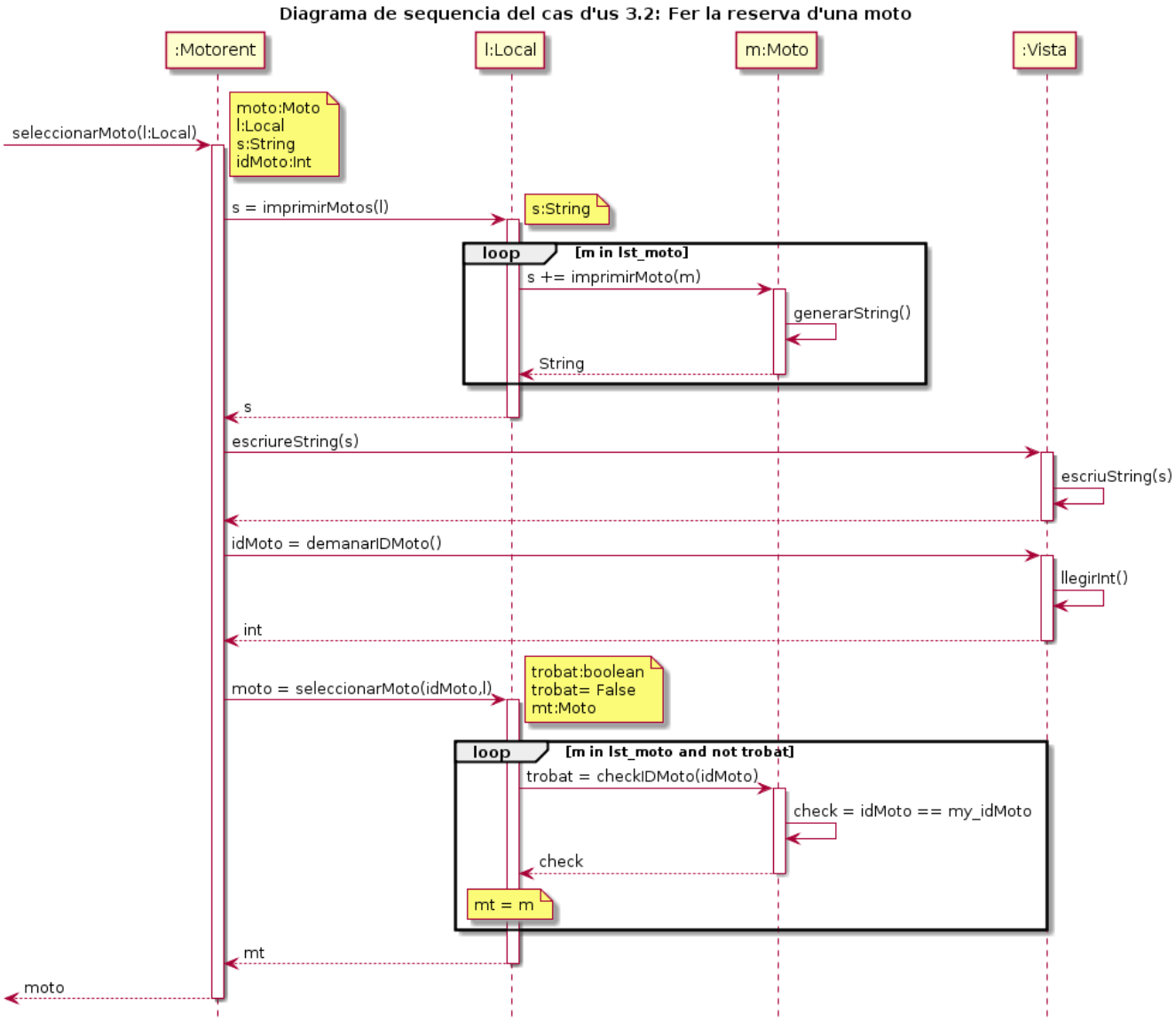


Diagrama de sequencia del cas d'us 3: Fer la reserva d'una moto
Client
:Motorent
cl:Client
origen:Local
motoR:Moto
:Vista
origen:Local
desti:local
dataRealitzacio:Data
dataRecollida: Data
dataDevolucio: Data
motoR:Moto
ferReserva()
r = crearReserva()
create
r:Reserva
origen = seleccionarLocal()
DS 3.1
desti = seleccionarLocal()
DS 3.1
setLocalOrigent(origen)
my_localOrigen = origen
setLocalDesti(desti)
my_localDesti = desti
motoR = seleccionarMoto(origen)
DS 3.2
setMotoReservada(motoR)
my_moto = motoR
setNoDisponible(motoR)
setNoDisponible(motoR)
my_Disponible = False
create
dataRecollida:Data
dia = demanarDia()
llegirInt()
Int
setDia(dia)
my_dia = dia
mes = demanarMes()
llegirInt()
Int
setMes(mes)
my_mes = mes
any = demanarAny()
llegirInt()
Int
setAny(any)
my_any = any
create
dataDevolucio:Data
dia = demanarDia()
llegirInt()
Int
setDia(dia)
my_dia = dia
mes = demanarMes()
llegirInt()
Int
setMes(mes)
my_mes = mes
any = demanarAny()
llegirInt()
Int
setAny(any)
my_any = any
dataRealitzacio = getDadaSistema()
LlegeixDataSistema()
Data
setDataRealitzacio(dataRealitzacio)
my_dataR = dataRealitzacio
setDataRecollida(dataRecollida)
my_dataRc = dataRecollida
setDataDevolucio(dataDevolucio)
my_dataD = dataDevolucio
codi = getCodi()
return my_Codi
my_codi
codi



Veure codi en PlantUML a l'apartat **Codis.**

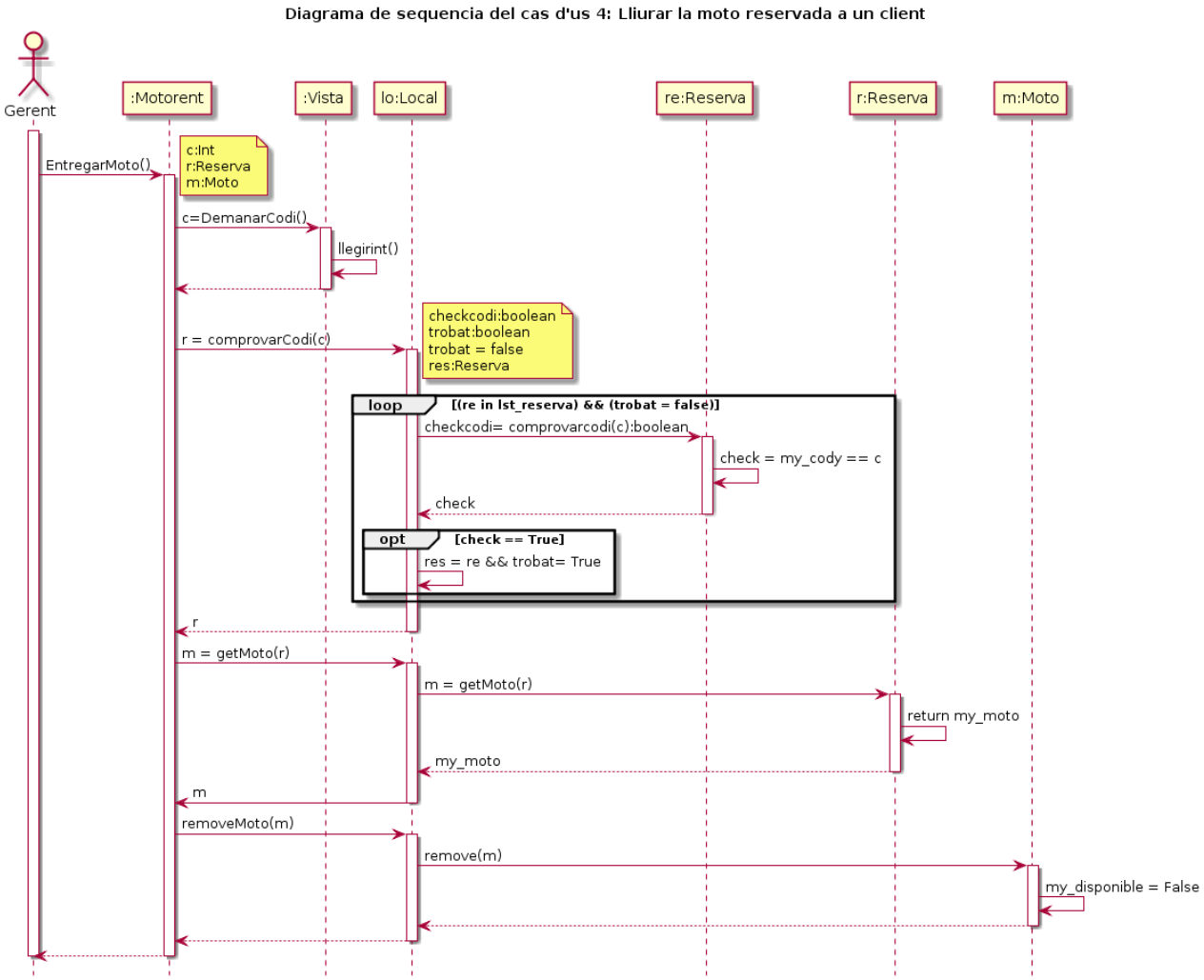
Aquest diagrama correspon al diagrama de seleccionar local (DS 3.1) on es mostra tots els locals disponibles i el client selecciona el local que ell desitja i es retorna el local escollit. Aquest diagrama està separat perquè el mètode d'escollir local l'utilitzarem en més d'un cas d'ús per tant es dissenyat a part per poder-lo utilitzar en el futur.



Veure codi en PlantUML a l'apartat **Codis.**

Aquest diagrama correspon al diagrama de seleccionar una moto del local escollit anteriorment (DS 3.2) on es mostra totes les motos disponibles en el local, en aquest cas en el local origen escollit pel client. Aquest diagrama està separat perquè el mètode d'escollir moto d'un local l'utilitzarem en més d'un cas d'ús per tant es dissenyat a part per poder-lo utilitzar en el futur.

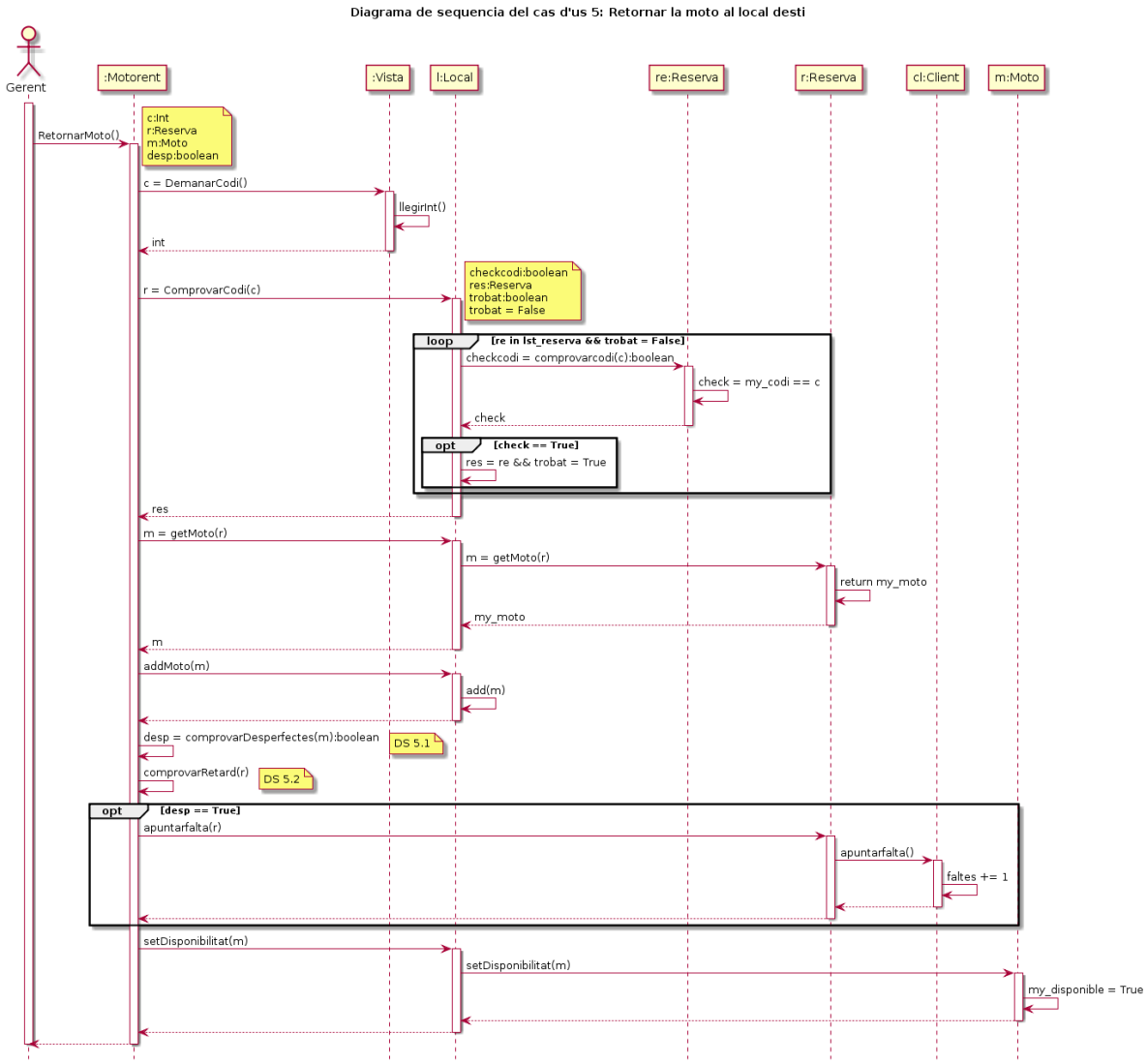
4.4. LLIURAR LA MOTO RESERVADA A UN CLIENT



Veure codi en PlantUML a l'apartat **Codis.**

Per poder lliurar la moto reservada al client primer de tot comprovem el codi de la reserva, si el codi és correcta obtenim la moto que havia reservat i un cop tenim la moto reservada l'eliminem del local origen ja que si entreguem la moto, la moto ja no es trobarà al local.

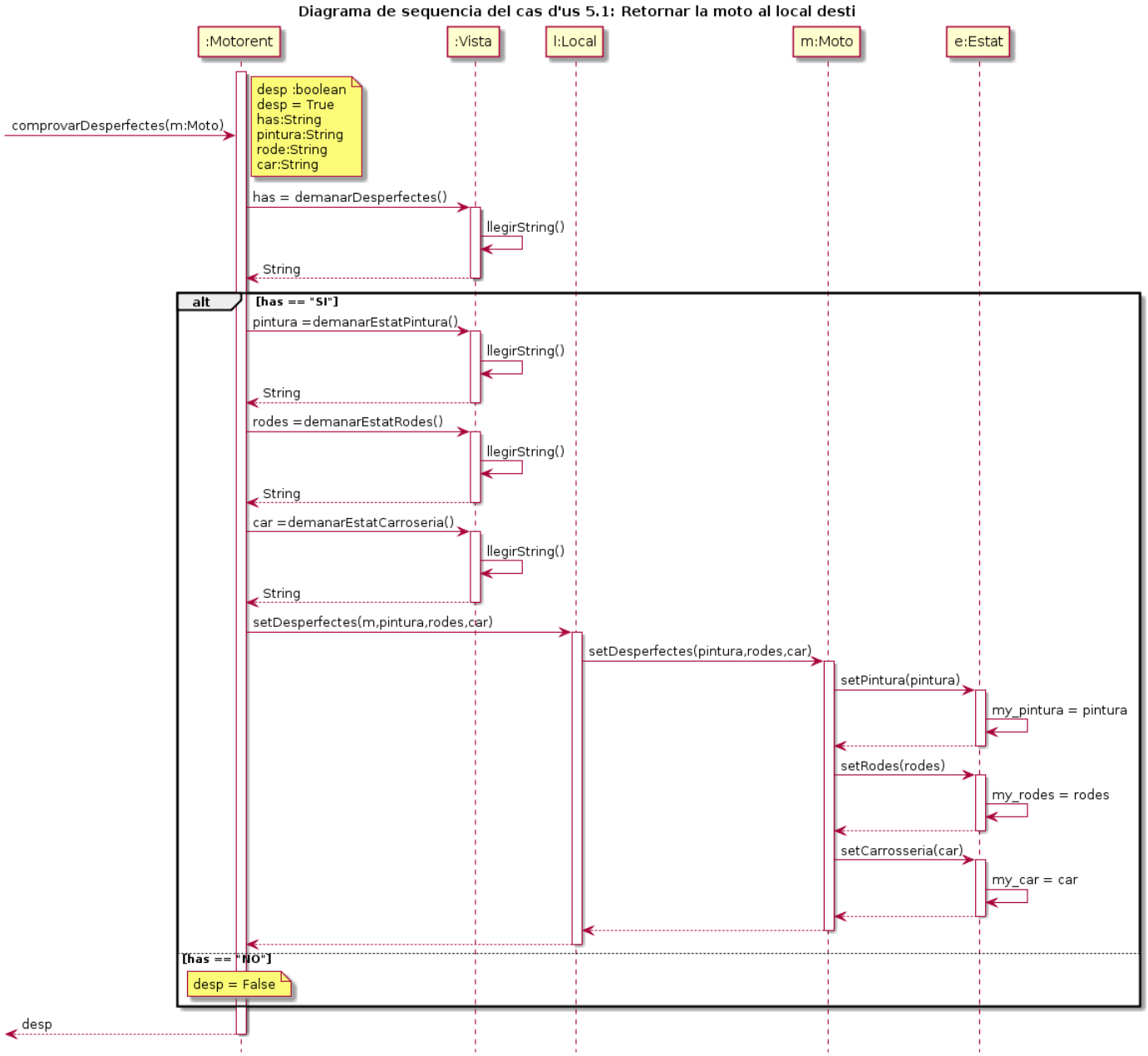
4.5. RETORNAR MOTO AL LOCAL DESTÍ



Veure codi en PlantUML a l'apartat **Codis.**

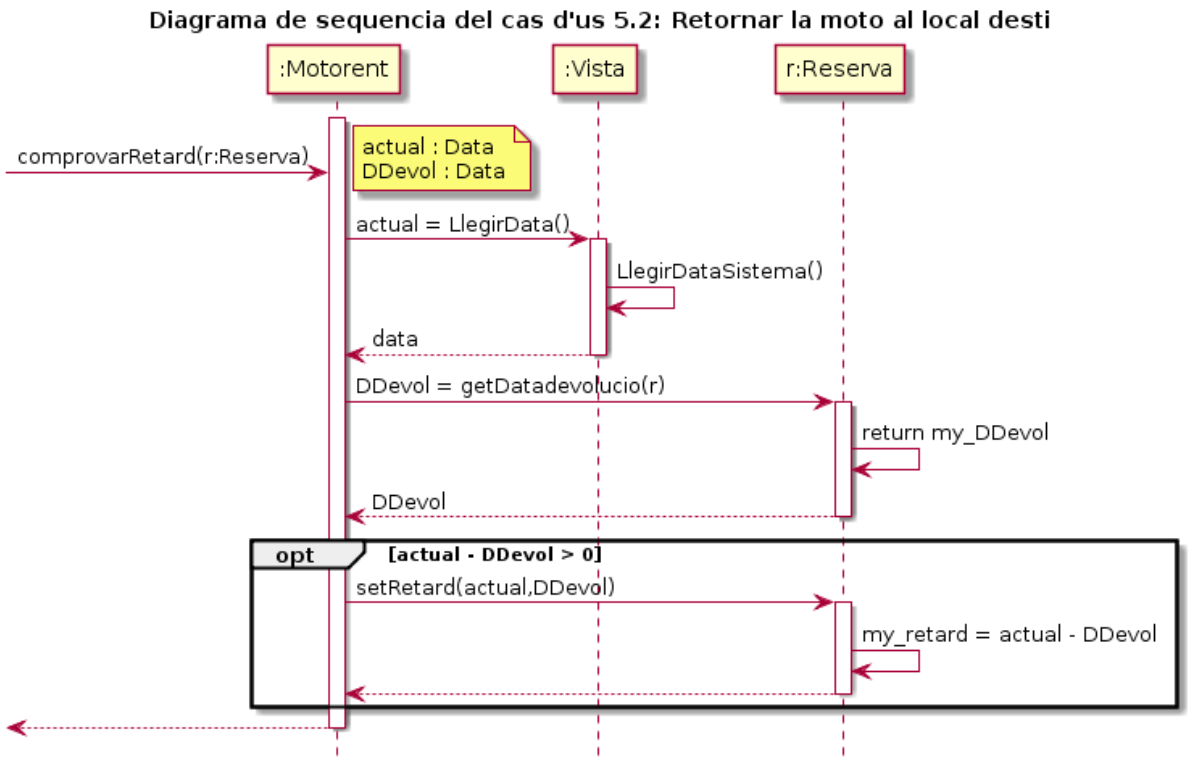
Per retornar la moto al local destí primer es torna a comprovar el codi, un cop el codi de la reserva és el correcte el gerent comprovarà l'estat de la moto (DS 5.1), si té algun desperfecte l'apuntarà, en cas de tenir desperfectes el gerent apuntarà una falta al client i afegirà la moto al local i la posarà a disponible. Un cop ha comprovat els desperfectes comprovarà si s'ha retornat en el període establert de la reserva, és a dir, que no s'ha retornat en retard.

En aquest cas només apuntarà una falta si ha estat retornada amb desperfectes, ja que l'enunciat no especifica que el retard conta com a falta.



Veure codi en PlantUML a l'apartat **Codis.**

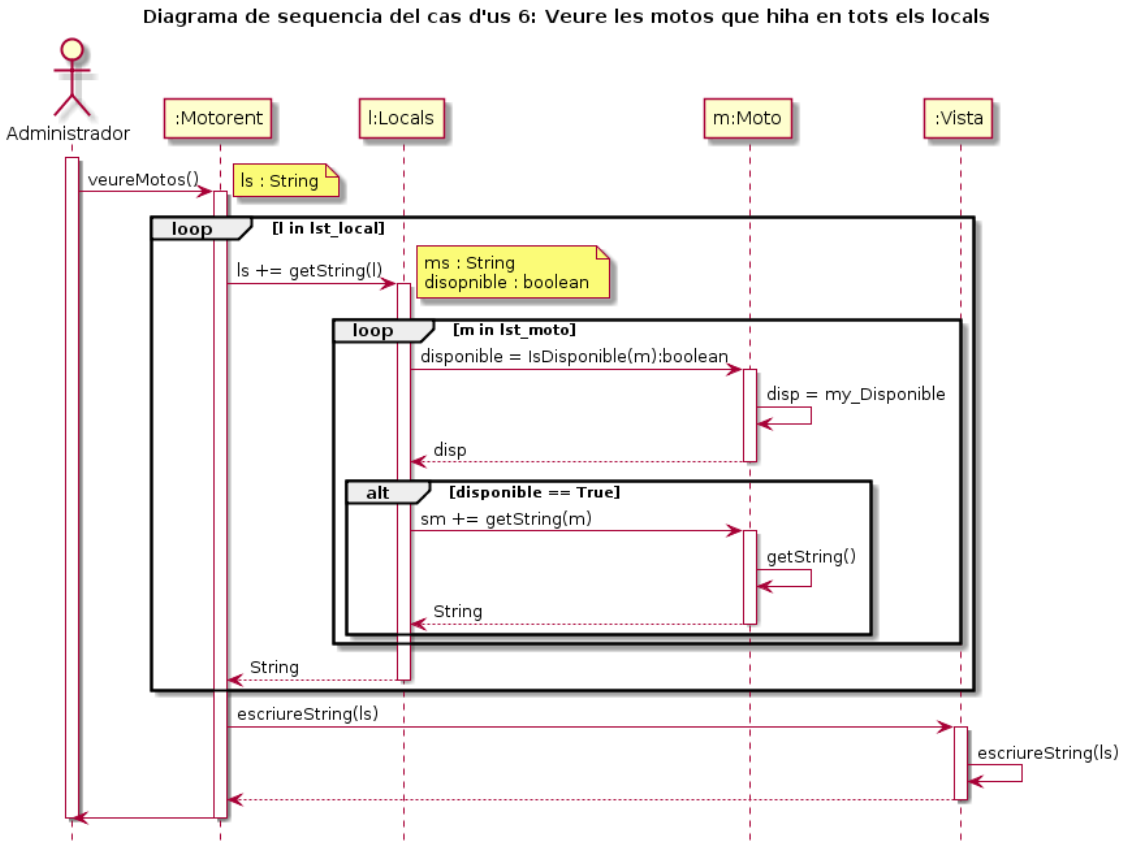
Aquest diagrama correspon al DS 5.1, comprovar desperfectes, el gerent comprova l'estat de la moto, és a dir, mira si té desperfectes, si en té apunta si són a les rodes, la pintura o la carrosseria.



Veure codi en PlantUML a l'apartat **Codis.**

Aquest diagrama correspon al DS 5.2, comprovar retard. El gerent llegeix la data del sistema, és a dir, al data en què s'ha retornat la moto i la compara amb la data de devolució de la reserva. Si la seva resta és més gran que 0 vol dir que hi ha hagut retard i es guarda el temps del retard en la reserva per posteriorment calcular el cost addicional al client.

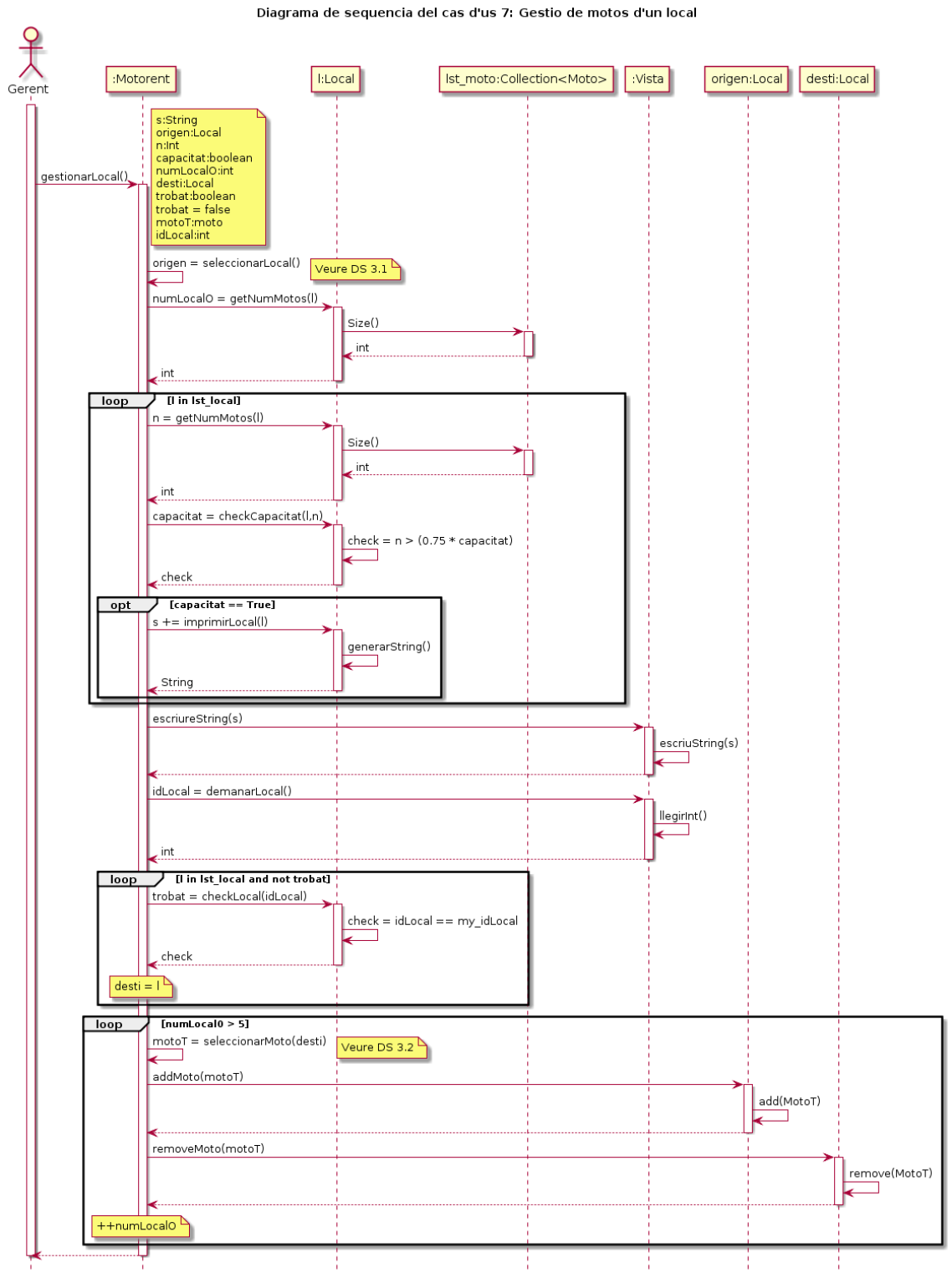
4.6. VEURE LES MOTOS QUE HI HA EN TOTS EL LOCALS



Veure codi en PlantUML a l'apartat **Codis.**

Per veure totes les motos que hi ha en tots els locals es fa un recorregut per cada local i per a cada moto que té es genera un string amb tota la informació que té la moto. Un cop generat aquest string es suma string amb la informació del local on hi ha les motos i s'imprimeix aquest string per pantalla. Aquesta acció només la pot realitzar l'administrador ja que és l'únic que té poder per poder accedir a tots als locals i veure les motos que tenen.

4.7. GESTÍO DE MOTOS D'UN LOCAL



Veure codi en PlantUML a l'apartat **Codis.**

Per tal que un gerent pugui gestionar les motos del seu local primer de tot es demana el local origen el qual es volen portar motos d'altres locals (veure DS 3.1). Un cop seleccionat el local s'obté el numero de motos d'aquest. A continuació es mostra per pantalla tots aquells locals que tenen més del 75% de la capacitat i es selecciona el local desitjat.

Un cop tenim els dos locals seleccionats és seleccionen les motos del local el qual s'agafen les motos (veure DS 3.2), s'afegeix la moto al local origen i s'elimina del local anterior. Aquest procés es realitzarà fins que el número de motos del local sigui més gran que el mínim, és a dir, quan tingui més de 5 motos ja no podrà moure'n més.

4.8. GENERAR INFORME AL FINAL DE CADA MES

Per tal de poder generar l'informe a cada mes primer de tot es demana a l'administrador que entri el mes del qual vol gener l'informe. Un cop tenim el mes per cada client del sistema es generarà un string de les reserves realitzades el mes escollit per l'administrador, és a dir, aquelles reserves que es van realitzar al mes escollit al principi. D'aquesta manera, s'imprimirà l'string i es mostrar per pantalla tota la informació de la reserva i el cost que se l'hi ha de cobrar. Nosaltres suposem que al imprimir una reserva ja es calcula el cost, ja que a reserva hi guardem el temps de retard de la reserva.

Veure codi en PlantUML a l'apartat **Codis.**

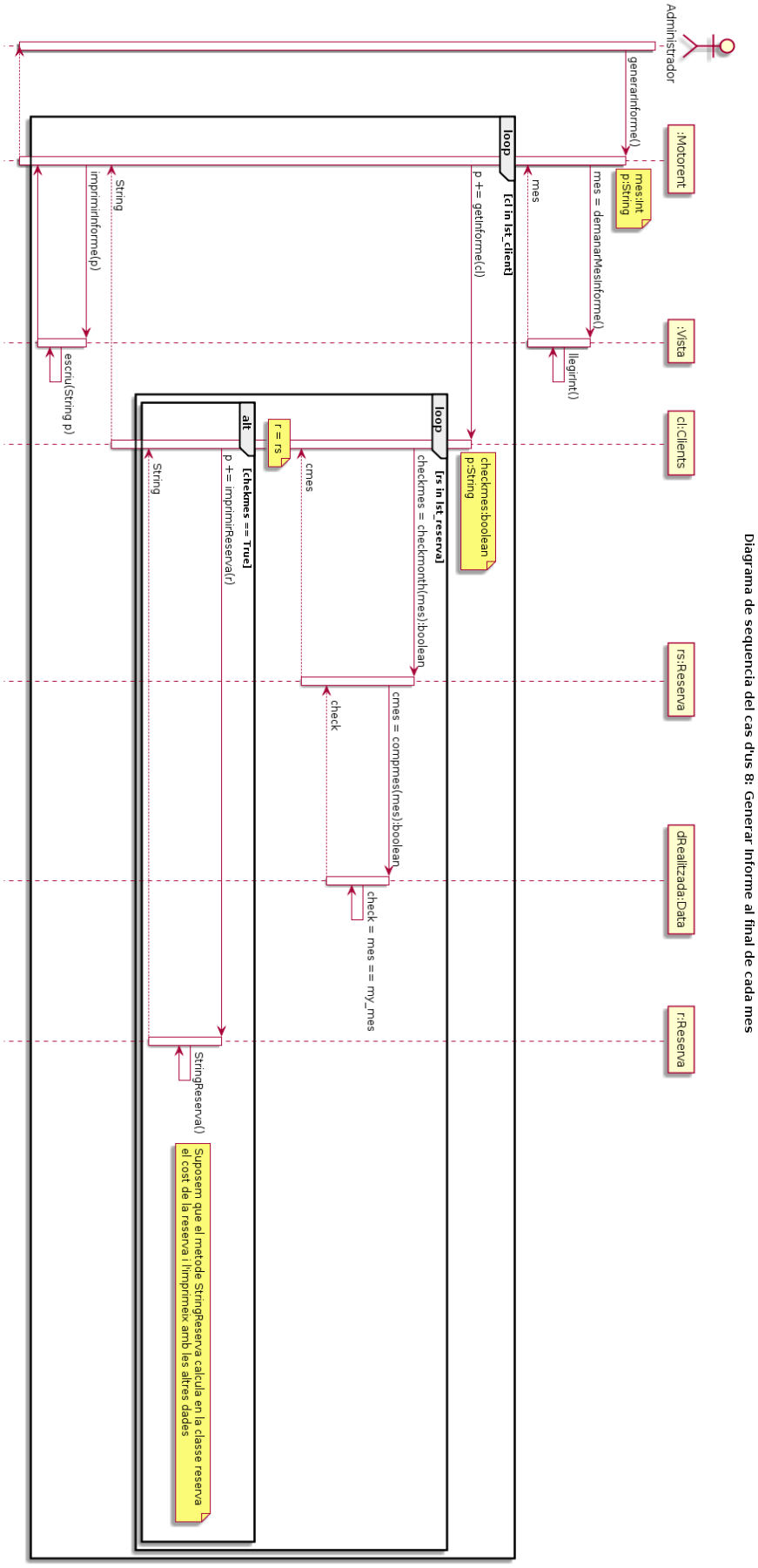
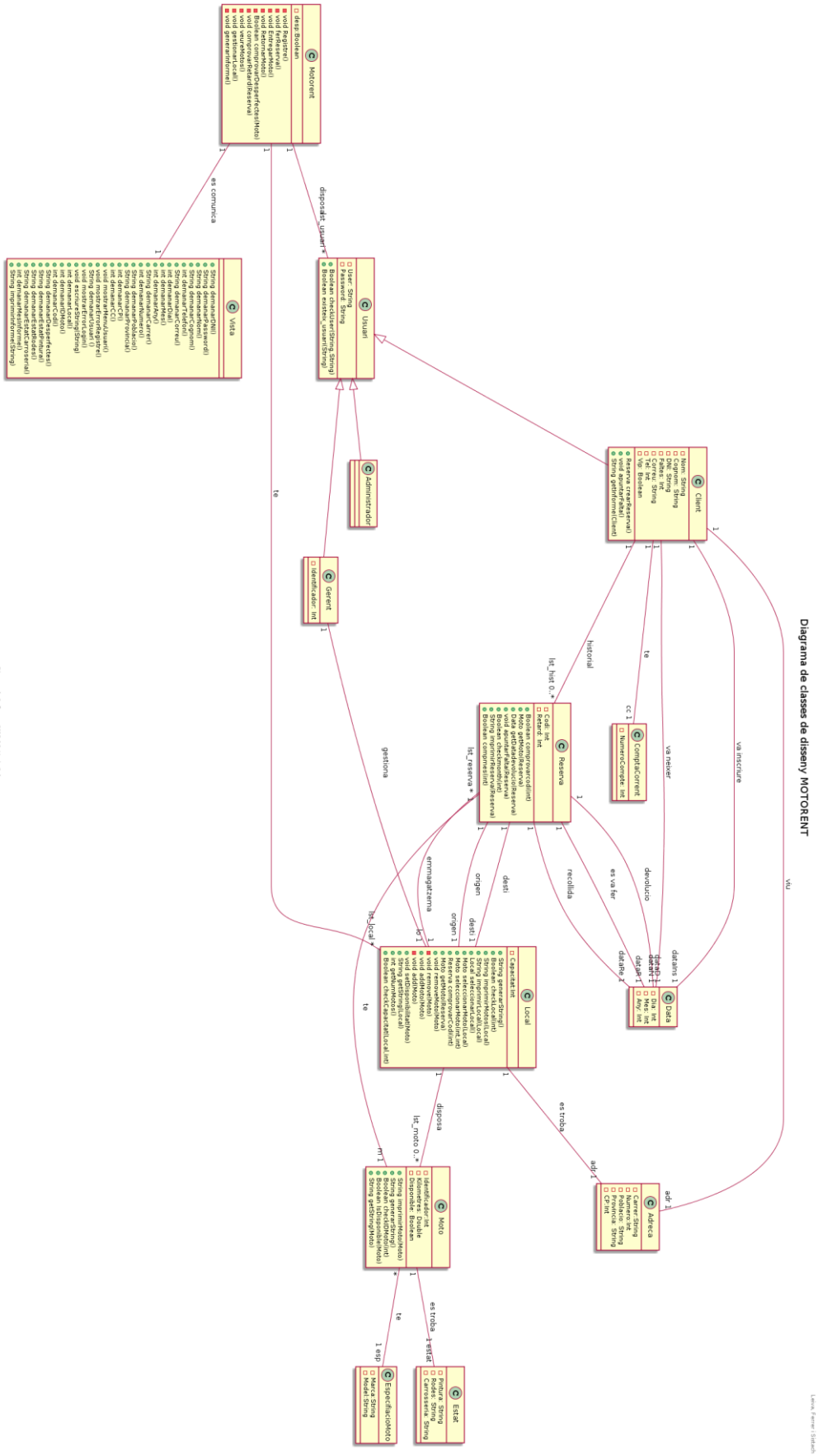


Diagrama de sequencia del cas d'us 8: Generar Informe al final de cada mes

5. MODEL DE CLASSES DE DISSENY



Veure codi en PlantUML a l'apartat **Codis.**

Com podem observar el diagrama de classes de disseny es molt semblant al de domini, de fet apareixen les mateixes mateixes classes que en el de domini la única diferència és que en aquest cas també apareixen els mètodes de cada classe i també apareix una nova classe que és la classe Vista que és l'encarregada de interactuar amb la persona, de demanar les dades i després comunicar-se amb el controlador. L'altre diferència respecte al model de domini és que en les connexions apareixen el nom de les variables que provenen d'altres classes.

En el diagrama no hem posat els setters ni els getters ja que es suposen que són necessaris i per tant no és necessari posar-los en el diagrama.

6. CONCLUSIONS

En aquesta primera pràctica de disseny de software hem après a fer un anàlisi de requisits donada una exigència del client. A partir de l'anàlisi de requisits també hem après a realitzar un diagrama de casos d'ús i fer els casos d'ús textuals. A continuació també ens hem format en el model de domini així com en el PlantUML a la hora de fer els diagrames corresponents. A m'és hem codificat a partir del Model de domini un petit projecte a NetBeans. Finalment hem après a crear diagrames de seqüència a partir dels casos d'ús i del model de domini que ens han permès començar definir les classes i els seus atributs en el projecte de NetBeans i que més endavant ens permetran crear els mètodes necessaris pel bon funcionament de l'aplicació.

7. DISTRIBUCIÓ DE LA FEINA I DECISIONS DE DISSENY

Les decisions de disseny en les que ens hem basat com a equip han sigut sempre de fer coses senzilles, però ben fetes i explicades correctament. També hem intentat fer-ho tot pensant en el codi final.

La distribució de la feina es la següent.

- **Alberto Leiva Cabello**: Correcció de la primera entrega, casos d'ús textuals, diagrames de seqüència, diagrama de classes de disseny i atributs de les classes del projecte NetBeans.
- **Marc Ferrer Margarit**: Correcció de la primera entrega, diagrames de seqüència, atributs de les classes del projecte NetBeans i informe.
- **Arnau Sistach Reinoso**: Correcció de la primera entrega, diagrama de seqüència i realització de la vista amb NetBeans.

8. CODIS

8.1. CASOS D'ÚS

```
@startuml
title Diagrama de casos d'ús de <b>MOTORENT</b>
header
Ferrer, Leiva, Sistach
endheader
center footer Disseny de Software 2015-16. Maria Salamo
left to right direction
skinparam packageStyle rect
actor Usuari as user <<Client>> #red
actor Usuarinoregistrat as notuser <<Usuari no registrat>>
actor Gerent as gerent #lightgreen
actor Administrador as administrador #lightblue
actor Temps as temps #blue
rectangle MOTORENT{
(1. Registrar-se) as registre
(2.Log In) as login
(3. Donar-se de baixa) as baja
(4. Reserva) as RES
(5. Modificar Local Destí) as MOD_LOCAL
(6.Entregar Moto) as ent_m
(7.Recollir Moto) as rec_m
(8.Gestionar Local) as reg_local
(9.Generar Informe Mensual) as gen_inf
(10.Veure motos de tots els locals) as ver_moto
(11.Escollir Local) as es_lc
(12.Client Vip) as cl_vip
'USUARI NO REGISTRAT'
notuser -> registre
'USUARI'
```

```
user -> login
user ->baja
user -> RES
user -> MOD_LOCAL
MOD_LOCAL <.. RES: extends
RES ..> es_lc: includes
MOD_LOCAL ..> es_lc: includes
'GERENT'
gerent -> ent_m
gerent -> rec_m
gerent -> reg_local
reg_local ..> es_lc: includes
'ADMINISTRADOR'
administrador -> gen_inf
administrador -> ver_moto
'TEMPS'
temps -> cl_vip
}
@enduml
```

8.2. MODEL DE DOMINI

```
@startuml
title  Model de Domini de MOTORENT
left to right direction
header
Leiva, Ferrer i Sistach
endheader
center footer Disseny de Software 2015-16. Maria Salamo
skinparam packageStyle rect
Usuari<|-- Client
Usuari<|-- Gerent
Usuari<|-- Administrador
Local "1"--"0..*" Moto: te
```

Gerent "1"--"1" Local: gestiona

Motorent "1"--"*" Usuari: disposa

Motorent "1"--"1" Usuari: treballa

Client "1"--"1" Adreca: viu

Client "1"--"1" Data: va neixer

Client "1"--"1" Data: es va inscriure

Client "1"--"1" ComptaCorrent: te

Motorent "1"--"*" Local: disposa

Local "1"--"1" Adreca: es troba

Reserva "1"--"1" Data: es fa

Reserva "1"--"1" Data: recollida

Reserva "1"--"1" Data: devolucio

Client "1"--"0..1" Reserva: realitza

Client "1"--"*" Reserva: historial

Reserva "0..1"--"1" Moto: te

Reserva "1"--"2" Local: conte

Reserva "*"--"1" Local: emmagatzema

Moto "1"--"1" Estat: es troba

Moto "*"--"1" EspecifiacioMoto: te

class Motorent

class Local{

-Capacitat:Int

}

class Adreca {

-Carrer:String

-Numero:Int

-Poblacio: String

-Provincia: String

-CP:Int

}

class Moto {

-Identificador:Int

-Kilometres: Double

-Disponible: Boolean

}

class Reserva {

-Codi: Int

-Retard: Int

}

class Data {

-Dia: Int

-Mes: Int

-Any: Int

}

class Estat {

-Pintura: String

-Rodes: String

-Carrosseria: String

}

class ComptaCorrent {

-NumeroCompta: Int

}

class Usuari {

-User: String

-Password: String

}

class Client {

-Nom: String

-Cognom: String

-DNI: String

-Faltes: Int

-Correu: String

-Tel: Int

-Vip: Boolean

}

class Gerent {

```
-Identificador: Int
}
class EspecifiacioMoto {
-Marca:String
-Model:String
}
@enduml
```

8.3. DIAGRAMA DE SEQÜÈNCIA 1

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 1: Registrar un usuari en el sistema
actor Persona as user
activate user
user ->":Motorent": Registre()
note right
        DNI:String
        u : Usuari
        existeix: boolean
end note
activate ":Motorent"
":Motorent" ->":Vista": DNI = demanarDNI()
activate ":Vista"
":Vista"-> ":Vista":llegirString()
":Motorent"<--":Vista" :DNI
deactivate ":Vista"
loop (us in lst_usuari) and (not existeix)
":Motorent" -> "us:Usuari": existeix := existeix_usuari(DNI):boolean
activate "us:Usuari"
"us:Usuari" -> "us:Usuari": check:= DNI == my_dni
"us:Usuari" --> ":Motorent": check
deactivate "us:Usuari"
end
```

```
alt existeix = false
create "u:Usuari"
":Motorent"->"u:Usuari" :create
activate "u:Usuari"
":Motorent"<--"u:Usuari":
deactivate "u:Usuari"
":Motorent"->":Motorent": demanarDades(u)
note right
        DS 1.1
end note
":Motorent"->":Motorent": guardarUsuari(u)
":Motorent" ->":Vista" : mostrarMenuUsuari()
activate ":Vista"
":Motorent" <-- ":Vista"
deactivate ":Vista"
else existeix == True
":Motorent" ->":Vista" : mostrarError Registre()
activate ":Vista"
":Motorent" <-- ":Vista"
deactivate ":Vista"
end
user <-- ":Motorent"
deactivate ":Motorent"
deactivate user
@enduml
```

8.4. DIAGRAMA DE SEQÜÈNCIA 1.1

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 1.1: Registrar un usuari en el sistema
activate ":Motorent"
->":Motorent":demanarDades(u:Usuari)
note over ":Motorent"
```

```
        d:Data
        direccio: Adreça
        cc:ComptaCorrent
        username: String
        password: String
        nom: String
        cognom:String
        DNI:String
        tel: Int
        correu: String
        dia: Int
        mes: Int
        any: Int
        carrer: String
        numero: Int
        poblacio: String
        provincia: String
        cp: Int
        numcc: Int
end note
":Motorent"->":Vista": username = demanarUser()
activate ":Vista"
":Vista"->":Vista": llegirString()
":Motorent"<--":Vista" :username
deactivate ":Vista"
":Motorent" -> "u:Usuari": setUserName(username)
activate "u:Usuari"
"u:Usuari"->"u:Usuari" : my_username = username
":Motorent"<-- "u:Usuari"
deactivate "u:Usuari"
":Motorent"->":Vista": password = demanarPassword()
activate ":Vista"
":Vista"->":Vista": llegirString()
```

":Motorent"<--":Vista" :password

deactivate ":Vista"

":Motorent" -> "u:Usuari": setPassword(password)

activate "u:Usuari"

"u:Usuari"->"u:Usuari" : my_password = password

":Motorent"<-- "u:Usuari"

deactivate "u:Usuari"

":Motorent"->":Vista": nom = demanarNom()

activate ":Vista"

":Vista"->":Vista": llegirString()

":Motorent"<--":Vista" :nom

deactivate ":Vista"

":Motorent" -> "u:Usuari": setNom(nom)

activate "u:Usuari"

"u:Usuari"->"u:Usuari" : my_nom = nom

":Motorent"<-- "u:Usuari"

deactivate "u:Usuari"

":Motorent"->":Vista": cognom = demanarCognom()

activate ":Vista"

":Vista"->":Vista": llegirString()

":Motorent"<--":Vista" :cognom

deactivate ":Vista"

":Motorent" -> "u:Usuari": setCognom(cognom)

activate "u:Usuari"

"u:Usuari"->"u:Usuari" : my_cognom = cognom

":Motorent"<-- "u:Usuari"

deactivate "u:Usuari"

":Motorent"->":Vista": DNI = demanarDNI()

activate ":Vista"

":Vista"->":Vista": llegirString()

":Motorent"<--":Vista" :DNI

deactivate ":Vista"

":Motorent"->"u:Usuari": setDNI(DNI)

activate "u:Usuari"

"u:Usuari"->"u:Usuari" : my_DNI = DNI

":Motorent"<-- "u:Usuari"

deactivate "u:Usuari"

":Motorent"->":Vista": tel = demanarTelefon()

activate ":Vista"

":Vista"->":Vista": llegirInt()

":Motorent"<--":Vista" :tel

deactivate ":Vista"

":Motorent" -> "u:Usuari": setTelefon(tel)

activate "u:Usuari"

"u:Usuari"->"u:Usuari" : my_tel = tel

":Motorent"<-- "u:Usuari"

deactivate "u:Usuari"

":Motorent"->":Vista": correu = demanarCorreu()

activate ":Vista"

":Vista"->":Vista": llegirString()

":Motorent"<--":Vista" :correu

deactivate ":Vista"

":Motorent" -> "u:Usuari": setCorreu(correu)

activate "u:Usuari"

"u:Usuari"->"u:Usuari" : my_correu = correu

":Motorent"<-- "u:Usuari"

deactivate "u:Usuari"

create "d:Data"

":Motorent"-> "d:Data" :create

activate "d:Data"

":Motorent"<--"d:Data":

deactivate "d:Data"

":Motorent"->":Vista": dia = demanarDia()

activate ":Vista"

":Vista"->":Vista": llegirInt()

":Motorent"<--":Vista" :dia

deactivate ":Vista"

":Motorent" -> "d:Data": setDia(dia)

activate "d:Data"

"d:Data"->"d:Data" : my_dia = dia

":Motorent"<-- "d:Data"

deactivate "d:Data"

":Motorent"->":Vista": mes = demanarMes()

activate ":Vista"

":Vista"->":Vista": llegirInt()

":Motorent"<--":Vista" :mes

deactivate ":Vista"

":Motorent" -> "d:Data": setMes(mes)

activate "d:Data"

"d:Data"->"d:Data" : my_mes = mes

":Motorent"<-- "d:Data"

deactivate "d:Data"

":Motorent"->":Vista": any = demanarAny()

activate ":Vista"

":Vista"->":Vista": llegirInt()

":Motorent"<--":Vista" :any

deactivate ":Vista"

":Motorent" -> "d:Data": setAny(any)

activate "d:Data"

"d:Data"->"d:Data" : my_any = any

":Motorent"<-- "d:Data"

deactivate "d:Data"

":Motorent" -> "u:Usuari": setDataN(d)

activate "u:Usuari"

"u:Usuari"->"u:Usuari" : my_data = d

":Motorent"<-- "u:Usuari"

deactivate "u:Usuari"

create "a:Adreça"

":Motorent"-> "a:Adreça" :create

activate "a:Adreça"

":Motorent"<--"a:Adreça":

deactivate "a:Adreça"

":Motorent"->":Vista": carrer = demanarCarrer()

activate ":Vista"

":Vista"->":Vista": llegirString()

":Motorent"<--":Vista" :carrer

deactivate ":Vista"

":Motorent" -> "a:Adreça": setCarrer(carrer)

activate "a:Adreça"

"a:Adreça"->"a:Adreça" : my_carrer = carrer

":Motorent"<-- "a:Adreça"

deactivate "a:Adreça"

":Motorent"->":Vista": numero = demanarNumero()

activate ":Vista"

":Vista"->":Vista": llegirInt()

":Motorent"<--":Vista" :numero

deactivate ":Vista"

":Motorent" -> "a:Adreça": setNumero(numero)

activate "a:Adreça"

"a:Adreça"->"a:Adreça" : my_numero = numero

":Motorent"<-- "a:Adreça"

deactivate "a:Adreça"

":Motorent"->":Vista": poblacio = demanarPoblacio()

activate ":Vista"

":Vista"->":Vista": llegirString()

":Motorent"<--":Vista" :poblacio

deactivate ":Vista"

":Motorent" -> "a:Adreça": setPoblacio(poblacio)

activate "a:Adreça"

"a:Adreça"->"a:Adreça" : my_poblacio = poblacio

":Motorent"<-- "a:Adreça"

deactivate "a:Adreça"

":Motorent"->":Vista": provincia = demanarProvincia()

activate ":Vista"

":Vista"->":Vista": llegirString()

":Motorent"<--":Vista" :provincia

deactivate ":Vista"

":Motorent" -> "a:Adreça": setProvincia(provincia)

activate "a:Adreça"

"a:Adreça"->"a:Adreça" : my_provincia = provincia

":Motorent"<-- "a:Adreça"

deactivate "a:Adreça"

":Motorent"->":Vista": cp = demanarCP()

activate ":Vista"

":Vista"->":Vista": llegirInt()

":Motorent"<--":Vista" :cp

deactivate ":Vista"

":Motorent" -> "a:Adreça": setCP(cp)

activate "a:Adreça"

"a:Adreça"->"a:Adreça" : my_cp = cp

":Motorent"<-- "a:Adreça"

deactivate "a:Adreça"

":Motorent" -> "u:Usuari": setDireccio(a)

activate "u:Usuari"

"u:Usuari"->"u:Usuari" : my_direccio = a

":Motorent"<-- "u:Usuari"

deactivate "u:Usuari"

create "cc:ComptaCorrent"

":Motorent"-> "cc:ComptaCorrent" :create

activate "cc:ComptaCorrent"

":Motorent"<--"cc:ComptaCorrent":

deactivate "cc:ComptaCorrent"

":Motorent"->":Vista": numcc = demanarNumCC()

activate ":Vista"

":Vista"->":Vista": llegirInt()

```
":Motorent"<--":Vista" :numcc
deactivate ":Vista"
":Motorent" -> "cc:ComptaCorrent": setNumCC(numcc)
activate "cc:ComptaCorrent"
"cc:ComptaCorrent"->"cc:ComptaCorrent" : my_numcc = numcc
":Motorent"<-- "cc:ComptaCorrent"
deactivate "cc:ComptaCorrent"
":Motorent" -> "u:Usuari": setComptaCorrent(cc)
activate "u:Usuari"
"u:Usuari"->"u:Usuari" : my_comptacorrent = cc
":Motorent"<-- "u:Usuari"
deactivate "u:Usuari"
<--":Motorent"
deactivate ":Motorent"
@enduml
```

8.5. DIAGRAMA DE SEQÜÈNCIA 2

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 2: Logar usuari en el sistema
actor Persona as user
activate user
user->":Motorent":login()
activate ":Motorent"
note right
   u : Usuari
   existeix : boolean
   existeix = False
   UserID = String
   password = String
end note
":Motorent" -> ":Vista": UserID = demanarUsuari()
activate ":Vista"
```

```
":Vista"->":Vista": llegirString()
":Motorent"<--":Vista": UserID
deactivate ":Vista"
":Motorent" ->":Vista": password = demanarPassword()
activate ":Vista"
":Vista"->":Vista": password = llegirString()
":Motorent"<--":Vista": password
deactivate ":Vista"
loop (us in lst_usuari) and (not existeix)
":Motorent" -> "us:Usuari": existeix := checkUser(UserID,password):boolean
activate "us:Usuari"
"us:Usuari" -> "us:Usuari": check:= UserID == my_UserID \n&&\n password == My_Password
"us:Usuari" --> ":Motorent": check
note over ":Motorent"
   u = us
end note
deactivate "us:Usuari"
end
alt existeix == True
alt u instanceof Client
":Motorent"->":Vista":mostrarMenuClient()
activate ":Vista"
":Motorent"<--":Vista"
deactivate ":Vista"
else u instance of Gerent
":Motorent"->":Vista":mostrarMenuGerent()
activate ":Vista"
":Motorent"<--":Vista"
deactivate ":Vista"
else u instance of Admin
":Motorent"->":Vista":mostrarMenuAdmin()
activate ":Vista"
":Motorent"<--":Vista"
```

```
deactivate ":Vista"
end
else
":Motorent"->":Vista":mostrarErrorLogin()
activate ":Vista"
":Vista"->":Vista": escriu(String("Usuari o password incorrecte"))
":Motorent"<--":Vista"
deactivate ":Vista"
end
":Motorent" --> user
deactivate ":Motorent"
deactivate user
@enduml
```

8.6. DIAGRAMA DE SEQÜÈNCIA 3

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 3: Fer la reserva d'una moto
actor Client as user
activate user
user ->":Motorent": ferReserva()
activate ":Motorent"
note right
   origen:Local
   desti:local
   dataRealitzacio:Data
   dataRecollida: Data
   dataDevolucio: Data
   motoR:Moto
end note
":Motorent"->"cl:Client":r = crearReserva()
activate "cl:Client"
create "r:Reserva"
```

```
"cl:Client"->"r:Reserva":create
activate "r:Reserva"
"cl:Client"<--"r:Reserva"
deactivate "r:Reserva"
":Motorent"<--"cl:Client"
deactivate "cl:Client"
":Motorent"->":Motorent":origen = seleccionarLocal()
note right
   DS 3.1
end note
":Motorent"->":Motorent":desti = seleccionarLocal()
note right
   DS 3.1
end note
":Motorent"->"r:Reserva": setLocalOrigent(origen)
activate "r:Reserva"
"r:Reserva"->"r:Reserva": my_localOrigen = origen
":Motorent"<--"r:Reserva"
deactivate "r:Reserva"
":Motorent"->"r:Reserva": setLocalDesti(desti)
activate "r:Reserva"
"r:Reserva"->"r:Reserva": my_localDesti = desti
":Motorent"<--"r:Reserva"
deactivate "r:Reserva"
":Motorent"->":Motorent":motoR = seleccionarMoto(origen)
note right
   DS 3.2
end note
":Motorent"->"r:Reserva": setMotoReservada(motoR)
activate "r:Reserva"
"r:Reserva"->"r:Reserva": my_moto = motoR
":Motorent"<--"r:Reserva"
deactivate "r:Reserva"
```

":Motorent"->"origen:Local": setNoDisponible(motoR)

activate "origen:Local"

"origen:Local"->"motoR:Moto": setNoDisponible(motoR)

activate "motoR:Moto"

"motoR:Moto"->"motoR:Moto": my_Disponible = False

"origen:Local"<--"motoR:Moto"

deactivate "motoR:Moto"

":Motorent"<--"origen:Local"

deactivate "origen:Local"

create "dataRecollida:Data"

":Motorent"-> "dataRecollida:Data": create

activate "dataRecollida:Data"

":Motorent"<-- "dataRecollida:Data"

deactivate "dataRecollida:Data"

":Motorent"->":Vista": dia = demanarDia()

activate ":Vista"

":Vista"->":Vista":llegirInt()

":Motorent"<--":Vista": Int

deactivate ":Vista"

":Motorent"-> "dataRecollida:Data": setDia(dia)

activate "dataRecollida:Data"

"dataRecollida:Data"->"dataRecollida:Data": my_dia = dia

":Motorent"<-- "dataRecollida:Data"

deactivate "dataRecollida:Data"

":Motorent"->":Vista": mes = demanarMes()

activate ":Vista"

":Vista"->":Vista":llegirInt()

":Motorent"<--":Vista": Int

deactivate ":Vista"

":Motorent"-> "dataRecollida:Data": setMes(mes)

activate "dataRecollida:Data"

"dataRecollida:Data"->"dataRecollida:Data": my_mes = mes

":Motorent"<-- "dataRecollida:Data"

deactivate "dataRecollida:Data"

":Motorent"->":Vista": any = demanarAny()

activate ":Vista"

":Vista"->":Vista":llegirInt()

":Motorent"<--":Vista": Int

deactivate ":Vista"

":Motorent"-> "dataRecollida:Data": setAny(any)

activate "dataRecollida:Data"

"dataRecollida:Data"->"dataRecollida:Data": my_any = any

":Motorent"<-- "dataRecollida:Data"

deactivate "dataRecollida:Data"

create "dataDevolucio:Data"

":Motorent"-> "dataDevolucio:Data": create

activate "dataDevolucio:Data"

":Motorent"<-- "dataDevolucio:Data"

deactivate "dataDevolucio:Data"

":Motorent"->":Vista": dia = demanarDia()

activate ":Vista"

":Vista"->":Vista":llegirInt()

":Motorent"<--":Vista": Int

deactivate ":Vista"

":Motorent"-> "dataDevolucio:Data": setDia(dia)

activate "dataDevolucio:Data"

"dataDevolucio:Data"->"dataDevolucio:Data": my_dia = dia

":Motorent"<-- "dataDevolucio:Data"

deactivate "dataDevolucio:Data"

":Motorent"->":Vista": mes = demanarMes()

activate ":Vista"

":Vista"->":Vista":llegirInt()

":Motorent"<--":Vista": Int

deactivate ":Vista"

":Motorent"-> "dataDevolucio:Data": setMes(mes)

activate "dataDevolucio:Data"

"dataDevolucio:Data"->"dataDevolucio:Data": my_mes = mes

":Motorent"<-- "dataDevolucio:Data"

deactivate "dataDevolucio:Data"

":Motorent"->":Vista": any = demanarAny()

activate ":Vista"

":Vista"->":Vista":llegirInt()

":Motorent"<--":Vista": Int

deactivate ":Vista"

":Motorent"-> "dataDevolucio:Data": setAny(any)

activate "dataDevolucio:Data"

"dataDevolucio:Data"->"dataDevolucio:Data": my_any = any

":Motorent"<-- "dataDevolucio:Data"

deactivate "dataDevolucio:Data"

":Motorent"->":Vista": dataRealitzacio = getDadaSistema()

activate ":Vista"

":Vista"->":Vista":LlegeixDataSistema()

":Motorent"<--":Vista": Data

deactivate ":Vista"

":Motorent"->"r:Reserva": setDataRealitzacio(dataRealitzacio)

activate "r:Reserva"

"r:Reserva"->"r:Reserva": my_dataR = dataRealitzacio

":Motorent"<--"r:Reserva"

deactivate "r:Reserva"

":Motorent"->"r:Reserva": setDataRecollida(dataRecollida)

activate "r:Reserva"

"r:Reserva"->"r:Reserva": my_dataRc = dataRecollida

":Motorent"<--"r:Reserva"

deactivate "r:Reserva"

":Motorent"->"r:Reserva": setDataDevolucio(dataDevolucio)

activate "r:Reserva"

"r:Reserva"->"r:Reserva": my_dataD = dataDevolucio

":Motorent"<--"r:Reserva"

deactivate "r:Reserva"

```
":Motorent"->"r:Reserva": codi = getCodi()
activate "r:Reserva"
"r:Reserva"->"r:Reserva": return my_Codi
":Motorent"<--"r:Reserva": my_codi
deactivate "r:Reserva"
user <--":Motorent": codi
deactivate ":Motorent"
deactivate user
@enduml
```

8.7. DIAGRAMA DE SEQÜÈNCIA 3.1

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 3.1: Fer la reserva d'una moto
->":Motorent":seleccionarLocal()
activate ":Motorent"
note right
   s:String
   idLocal:int
   trobat:boolean
   trobat = False
   local = Local
end note
loop l in lst_local
":Motorent" -> "l:Local": s += imprimirLocal(l)
activate "l:Local"
"l:Local"->"l:Local":generarString()
":Motorent"<--"l:Local": String
deactivate "l:Local"
end loop
":Motorent"->":Vista": escriureString(s)
activate ":Vista"
":Vista"->":Vista":escriuString(s)
```

```
":Motorent"<--":Vista"
deactivate ":Vista"
":Motorent"->":Vista": idLocal = demanarLocal()
activate ":Vista"
":Vista"->":Vista":llegirInt()
":Motorent"<--":Vista": int
deactivate ":Vista"
loop l in lst_local and not trobat
":Motorent"->"l:Local": trobat = checkLocal(idLocal)
activate "l:Local"
"l:Local"->"l:Local": check = idLocal == my_idLocal
":Motorent"<--"l:Local": check
deactivate "l:Local"
note over ":Motorent"
   local = l
end note
end loop
<--":Motorent": local
deactivate ":Motorent"
@enduml
```

8.8. DIAGRAMA DE SEQÜÈNCIA 3.2

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 3.2: Fer la reserva d'una moto
->":Motorent":seleccionarMoto(l:Local)
activate ":Motorent"
note right
   moto:Moto
   l:Local
   s:String
   idMoto:Int
end note
```

```
":Motorent"->"l:Local": s = imprimirMotos(l)
activate "l:Local"
note right
   s:String
end note
loop m in lst_moto
"l:Local" -> "m:Moto": s += imprimirMoto(m)
activate "m:Moto"
"m:Moto"->"m:Moto":generarString()
"l:Local"<--"m:Moto": String
deactivate "m:Moto"
end loop
":Motorent"<--"l:Local": s
deactivate "l:Local"
":Motorent"->":Vista": escriureString(s)
activate ":Vista"
":Vista"->":Vista":escriuString(s)
":Motorent"<--":Vista"
deactivate ":Vista"
":Motorent"->":Vista": idMoto = demanarIDMoto()
activate ":Vista"
":Vista"->":Vista":llegirInt()
":Motorent"<--":Vista": int
deactivate ":Vista"
":Motorent"->"l:Local": moto = seleccionarMoto(idMoto,l)
activate "l:Local"
note right
   trobat:boolean
   trobat= False
   mt:Moto
end note
loop m in lst_moto and not trobat
"l:Local"->"m:Moto": trobat = checkIDMoto(idMoto)
```

```
activate "m:Moto"
"m:Moto"->"m:Moto": check = idMoto == my_idMoto
"l:Local"<--"m:Moto":check
deactivate "m:Moto"
note over "l:Local"
   mt = m
end note
end loop
":Motorent"<--"l:Local": mt
deactivate "l:Local"
<--":Motorent": moto
deactivate ":Motorent"
@enduml
```

8.9. DIAGRAMA DE SEQÜÈNCIA 4

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 4: Lliurar la moto reservada a un client
actor Gerent as user
activate user
user ->":Motorent": EntregarMoto()
activate ":Motorent"
note right
        c:Int
        r:Reserva
        m:Moto
end note
":Motorent"->":Vista": c=DemanarCodi()
activate ":Vista"
":Vista" -> ":Vista" : llegirint()
":Vista" --> ":Motorent"
deactivate ":Vista"
":Motorent" -> "lo:Local": r = comprovarCodi(c)
```

```
activate "lo:Local"
note right
        checkcodi:boolean
        trobat:boolean
        trobat = false
        res:Reserva
end note
loop (re in lst_reserva) && (trobat = false)
"lo:Local" -> "re:Reserva" : checkcodi= comprovarcodi(c):boolean
activate "re:Reserva"
"re:Reserva" -> "re:Reserva" : check = my_cody == c
"re:Reserva" --> "lo:Local" : check
deactivate "re:Reserva"
opt check == True
"lo:Local" -> "lo:Local": res = re && trobat= True
end
end loop
"lo:Local" --> ":Motorent" : r
deactivate "lo:Local"
":Motorent"->"lo:Local":m = getMoto(r)
activate "lo:Local"
"lo:Local" ->"r:Reserva": m = getMoto(r)
activate "r:Reserva"
"r:Reserva" -> "r:Reserva": return my_moto
"r:Reserva"-->"lo:Local": my_moto
deactivate "r:Reserva"
"lo:Local"->":Motorent": m
deactivate "lo:Local"
":Motorent"-> "lo:Local": removeMoto(m)
activate "lo:Local"
"lo:Local"->"lo:Local": remove(m)
":Motorent"<--"lo:Local"
deactivate "lo:Local"
```

```
":Motorent" --> user
deactivate ":Motorent"
deactivate user
@enduml
```

8.10. DIAGRAMA DE SEQÜÈNCIA 5

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 5: Retornar la moto al local desti
actor Gerent as user
activate user
user->":Motorent": RetornarMoto()
activate ":Motorent"
note right
        c:Int
        r:Reserva
        m:Moto
        desp:boolean
end note
":Motorent"->":Vista": c = DemanarCodi()
activate ":Vista"
":Vista"->":Vista": llegirInt()
":Motorent"<--":Vista": int
deactivate ":Vista"
":Motorent"-> "l:Local": r = ComprovarCodi(c)
activate "l:Local"
note right
        checkcodi:boolean
        res:Reserva
        trobat:boolean
        trobat = False
end note
loop re in lst_reserva && trobat = False
```

```
"l:Local"->"re:Reserva": checkcodi = comprovarcodi(c):boolean
activate "re:Reserva"
"re:Reserva"->"re:Reserva": check = my_codi == c
"l:Local"<-- "re:Reserva": check
deactivate "re:Reserva"
opt check == True
"l:Local" -> "l:Local": res = re && trobat = True
end
end loop
":Motorent"<--"l:Local": res
deactivate "l:Local"
":Motorent"-> "l:Local": m = getMoto(r)
activate "l:Local"
"l:Local"-> "r:Reserva": m = getMoto(r)
activate "r:Reserva"
"r:Reserva"->"r:Reserva": return my_moto
"l:Local"<-- "r:Reserva": my_moto
deactivate "r:Reserva"
":Motorent"<--"l:Local": m
deactivate "l:Local"
":Motorent"-> "l:Local": addMoto(m)
activate "l:Local"
"l:Local"->"l:Local": add(m)
":Motorent"<--"l:Local"
deactivate "l:Local"
":Motorent"->":Motorent": desp = comprovarDesperfectes(m):boolean
note right
   DS 5.1
end note
":Motorent"->":Motorent" :comprovarRetard(r)
note right
   DS 5.2
end note
```

```
opt desp == True
":Motorent"-> "r:Reserva" : apuntarfalta(r)
activate "r:Reserva"
"r:Reserva"->"cl:Client": apuntarfalta()
activate "cl:Client"
"cl:Client"->"cl:Client": faltes += 1
"r:Reserva"<--"cl:Client"
deactivate "cl:Client"
":Motorent"<--"r:Reserva"
deactivate "r:Reserva"
end
":Motorent"->"l:Local": setDisponibilitat(m)
activate "l:Local"
"l:Local"-> "m:Moto": setDisponibilitat(m)
activate "m:Moto"
"m:Moto"->"m:Moto": my_disponible = True
"l:Local"<--"m:Moto"
deactivate "m:Moto"
":Motorent"<--"l:Local"
deactivate "l:Local"
user <--":Motorent"
deactivate ":Motorent"
deactivate user
@enduml
```

8.11. DIAGRAMA DE SEQÜÈNCIA 5.1

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 5.1: Retornar la moto al local desti
activate ":Motorent"
->":Motorent": comprovarDesperfectes(m:Moto)
note right
        desp :boolean
```

```
        desp = True
        has:String
        pintura:String
        rode:String
        car:String
end note
":Motorent" -> ":Vista": has = demanarDesperfectes()
activate ":Vista"
":Vista"-> ":Vista": llegirString()
":Motorent"<-- ":Vista": String
deactivate ":Vista"
alt has == "SI"
":Motorent"-> ":Vista": pintura =demanarEstatPintura()
activate ":Vista"
":Vista"-> ":Vista": llegirString()
":Motorent"<-- ":Vista": String
deactivate ":Vista"
":Motorent"-> ":Vista": rodes =demanarEstatRodes()
activate ":Vista"
":Vista"-> ":Vista": llegirString()
":Motorent"<-- ":Vista": String
deactivate ":Vista"
":Motorent"-> ":Vista": car =demanarEstatCarroseria()
activate ":Vista"
":Vista"-> ":Vista": llegirString()
":Motorent"<-- ":Vista": String
deactivate ":Vista"
":Motorent"->"l:Local": setDesperfectes(m,pintura,rodes,car)
activate "l:Local"
"l:Local"->"m:Moto": setDesperfectes(pintura,rodes,car)
activate "m:Moto"
"m:Moto"->"e:Estat": setPintura(pintura)
activate "e:Estat"
```

```
"e:Estat"->"e:Estat": my_pintura = pintura
"m:Moto"<--"e:Estat"
deactivate "e:Estat"
"m:Moto"->"e:Estat": setRodes(rodes)
activate "e:Estat"
"e:Estat"->"e:Estat": my_rodes = rodes
"m:Moto"<--"e:Estat"
deactivate "e:Estat"
"m:Moto"->"e:Estat": setCarrosseria(car)
activate "e:Estat"
"e:Estat"->"e:Estat": my_car = car
"m:Moto"<--"e:Estat"
deactivate "e:Estat"
"l:Local"<--"m:Moto"
deactivate "m:Moto"
":Motorent"<--"l:Local"
deactivate "l:Local"
else has == "NO"
note over ":Motorent"
        desp = False
end note
end
<--":Motorent": desp
deactivate ":Motorent"
@enduml
```

8.12. DIAGRAMA DE SEQÜÈNCIA 5.2

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 5.2: Retornar la moto al local desti
activate ":Motorent"
->":Motorent":comprovarRetard(r:Reserva)
note right
```

```
   actual : Data
   DDevol : Data
end note
":Motorent"-> ":Vista": actual = LlegirData()
activate ":Vista"
":Vista"->":Vista": LlegirDataSistema()
":Motorent"<--":Vista": data
deactivate ":Vista"
":Motorent"->"r:Reserva": DDevol = getDatadevolucio(r)
activate "r:Reserva"
"r:Reserva"->"r:Reserva": return my_DDevol
":Motorent"<--"r:Reserva" :DDevol
deactivate "r:Reserva"
opt actual - DDevol > 0
":Motorent"-> "r:Reserva": setRetard(actual,DDevol)
activate "r:Reserva"
"r:Reserva"->"r:Reserva": my_retard = actual - DDevol
":Motorent"<--"r:Reserva"
deactivate "r:Reserva"
end
<--":Motorent"
deactivate ":Motorent"
@enduml
```

8.13. DIAGRAMA DE SEQÜÈNCIA 6

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 6: Veure les motos que hiha en tots els locals
actor Administrador as user
activate user
user ->":Motorent": veureMotos()
activate ":Motorent"
note right
```

```
   ls : String
end note
loop l in lst_local
":Motorent"-> "l:Locals": ls += getString(l)
activate "l:Locals"
note right
   ms : String
   disopnible : boolean
end note
loop m in lst_moto
"l:Locals"->"m:Moto": disponible = IsDisponible(m):boolean
activate "m:Moto"
"m:Moto"->"m:Moto":disp = my_Disponible
"l:Locals"<--"m:Moto": disp
deactivate "m:Moto"
alt disponible == True
"l:Locals"->"m:Moto": sm += getString(m)
activate "m:Moto"
"m:Moto"->"m:Moto": getString()
"l:Locals"<-- "m:Moto": String
deactivate "m:Moto"
end
end loop
":Motorent"<--"l:Locals": String
deactivate "l:Locals"
end loop
":Motorent" -> ":Vista": mostrar(ls)
activate ":Vista"
":Vista"->":Vista": escriureString(ls)
":Motorent"<--":Vista"
deactivate ":Vista"
user <- ":Motorent"
deactivate ":Motorent"
```

```
deactivate user
@enduml
```

8.14. DIAGRAMA DE SEQÜÈNCIA 7

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 7: Gestio de motos d'un local
actor Gerent as user
activate user
user -> ":Motorent": gestionarLocal()
activate ":Motorent"
note right
   s:String
   origen:Local
   n:Int
   capacitat:boolean
   numLocalO:int
   desti:Local
   trobat:boolean
   trobat = false
   motoT:moto
   idLocal:int
end note
":Motorent"->":Motorent":origen = seleccionarLocal()
note right
   Veure DS 3.1
end note
":Motorent"->"l:Local": numLocalO = getNumMotos(l)
activate "l:Local"
"l:Local"->"lst_moto:Collection<Moto>": Size()
activate "lst_moto:Collection<Moto>"
"l:Local"<--"lst_moto:Collection<Moto>": int
deactivate "lst_moto:Collection<Moto>"
```

"l:Local"-->":Motorent": int

deactivate "l:Local"

loop l in lst_local

":Motorent"->"l:Local": n = getNumMotos(l)

activate "l:Local"

"l:Local"->"lst_moto:Collection<Moto>": Size()

activate "lst_moto:Collection<Moto>"

"l:Local"<--"lst_moto:Collection<Moto>": int

deactivate "lst_moto:Collection<Moto>"

"l:Local"-->":Motorent": int

deactivate "l:Local"

":Motorent"->"l:Local": capacitat = checkCapacitat(l,n)

activate "l:Local"

"l:Local"->"l:Local": check = n > (0.75 * capacitat)

":Motorent"<--"l:Local": check

deactivate "l:Local"

opt capacitat == True

":Motorent"->"l:Local": s += imprimirLocal(l)

activate "l:Local"

"l:Local"->"l:Local": generarString()

":Motorent"<--"l:Local": String

deactivate "l:Local"

end

end loop

":Motorent" ->":Vista": escriureString(s)

activate ":Vista"

":Vista"->":Vista" : escriuString(s)

":Motorent"<--":Vista"

deactivate ":Vista"

":Motorent" ->":Vista": idLocal = demanarIDLocal()

activate ":Vista"

":Vista"->":Vista" : llegirInt()

```
":Motorent"<--":Vista": int
deactivate ":Vista"
loop l in lst_local and not trobat
":Motorent"->"l:Local": trobat = checkLocal(idLocal)
activate "l:Local"
"l:Local"->"l:Local": check = idLocal == my_idLocal
":Motorent"<--"l:Local": check
deactivate "l:Local"
note over ":Motorent"
   desti = l
end note
end loop
loop numLocal0 > 5
":Motorent"->":Motorent":motoT = seleccionarMoto(desti)
note right
   Veure DS 3.2
end note
":Motorent"->"origen:Local": addMoto(motoT)
activate "origen:Local"
"origen:Local"->"origen:Local": add(MotoT)
":Motorent"<--"origen:Local"
deactivate "origen:Local"
":Motorent"->"desti:Local": removeMoto(motoT)
activate "desti:Local"
"desti:Local"->"desti:Local": remove(MotoT)
":Motorent"<--"desti:Local"
deactivate "desti:Local"
note over ":Motorent"
   ++numLocalO
end note
end loop
user <-- ":Motorent"
deactivate ":Motorent"
```

```
deactivate user
@enduml
```

8.15. DIAGRAMA DE SEQÜÈNCIA 8

```
@startuml
hide footbox
title Diagrama de sequencia del cas d'us 8: Generar Informe al final de cada mes
actor Administrador as user
activate user
user -> ":Motorent": generarInforme()
activate ":Motorent"
note right
        mes:Int
        p:String
end note
":Motorent"->":Vista":mes = demanarMesInforme()
activate ":Vista"
":Vista"->":Vista": llegirInt()
":Motorent"<--":Vista": mes
deactivate ":Vista"
loop cl in lst_client
":Motorent"->"cl:Clients" : p += getInforme(cl)
activate "cl:Clients"
note right
        checkmes:boolean
        p:String
end note
loop rs in lst_reserva
"cl:Clients"->"rs:Reserva": checkmes = checkmonth(mes):boolean
activate "rs:Reserva"
"rs:Reserva"-> "dRealitzada:Data": cmes = compmes(mes):boolean
activate "dRealitzada:Data"
"dRealitzada:Data"->"dRealitzada:Data": check = mes == my_mes
"rs:Reserva"<--"dRealitzada:Data": check
```

```
deactivate "dRealitzada:Data"
"cl:Clients"<--"rs:Reserva":cmes
deactivate "rs:Reserva"
note over "cl:Clients"
   r = rs
end note
alt chekmes == True
"cl:Clients"->"r:Reserva":p += imprimirReserva(r)
activate "r:Reserva"
"r:Reserva"-> "r:Reserva" : StringReserva()
note right
        Suposem que el metode StringReserva calcula en la classe reserva
        el cost de la reserva i l'imprimeix amb les altres dades
end note
"cl:Clients"<-- "r:Reserva" : String
deactivate "r:Reserva"
end
end loop
":Motorent"<--"cl:Clients" : String
deactivate "cl:Clients"
":Motorent" ->":Vista": imprimirInforme(p)
activate ":Vista"
":Vista" -> ":Vista" : escriu(String p)
":Vista" -> ":Motorent"
deactivate ":Vista"
end loop
user <--":Motorent"
deactivate ":Motorent"
deactivate user
@enduml
```

8.16. DIAGRAMA DE CLASSES DE DISSENY

```
@startuml
```

```
title  Diagrama de classes de disseny MOTORENT
left to right direction
header
Leiva, Ferrer i Sistach
endheader
center footer Disseny de Software 2015-16. Maria Salamo
skinparam packageStyle rect
Usuari<|--- Client
Usuari<|--- Gerent
Usuari<|--- Administrador
class Vista{
+String demanarDNI()
+String demanarPassword()
+String demanarNom()
+String demanarCognom()
+int demanarTelefon()
+String demanarCorreu()
+int demanarDia()
+int demanarMes()
+int demanarAny()
+String demanarCarrer()
+int demanarNumero()
+String demanarPoblacio()
+String demanarProvincia()
+int demanarCP()
+int demanarCC()
+void mostrarMenuUsuari()
+void mostrarErrorRegistre()
+String demanarUsuari ()
+void mostrarErrorLogin()
+void escriureString(String)
+int demanarLocal()
+int demanarIDMoto()
```

+int demanarCodi()

+String demanarDesperfectes()

+String demanarEstatPintura()

+String demanarEstatRodes()

+String demanarEstatCarroseria()

+int demanarMesInforme()

+String imprimirInforme(String)

}

class Motorent{

-desp:Boolean

-void Registre()

-void ferReserva()

-void EntregarMoto()

-void RetornarMoto()

-Boolean comprovarDesperfectes(Moto)

-void comprovarRetard(Reserva)

-void veureMotos()

-void gestionarLocal()

-void generarInforme()

}

class Local{

-Capacitat:Int

+ String generarString()

+ Boolean checkLocal(int)

+ String imprimirMotos(Local)

+ String imprimirLocal(Local)

+ Local seleccionarLocal()

+ Moto seleccionarMoto(Local)

+ Moto seleccionarMoto(int,int)

+ Reserva comprovarCodi(int)

+ Moto getMoto(Reserva)

+ void removeMoto(Moto)

- void remove(Moto)

+ void addMoto(Moto)

- void add(Moto)

+ void setDisponibilitat(Moto)

+ String getString(Local)

+ int getNumMotos()

+ Boolean checkCapacitat(Local,int)

}

class Adreca {

-Carrer:String

-Numero:Int

-Poblacio: String

-Provincia: String

-CP:Int

}

class Moto {

-Identificador:Int

-Kilometres: Double

-Disponible: Boolean

+String imprimirMoto(Moto)

+String generarString()

+Boolean checkIDMoto(int)

+Boolean IsDisponible(Moto)

+String getString(Moto)

}

class Reserva {

-Codi: Int

-Retard: Int

+Boolean comprovarcodi(int)

+Moto getMoto(Reserva)

+Data getDatadevolucio(Reserva)

+void apuntarFalta(Reserva)

+Boolean checkmonth(int)

+String imprimirReserva(Reserva)

+Boolean compmes(int)

}

class Data {

-Dia: Int

-Mes: Int

-Any: Int

}

class Estat {

-Pintura: String

-Rodes: String

-Carrosseria: String

}

class CompteCorrent {

-NumeroCompte: Int

}

class Usuari {

-User: String

-Password: String

+Boolean checkUser(String,String)

+Boolean existeix_usuari(String)

}

class Client {

-Nom: String

-Cognom: String

-DNI: String

-Faltes: Int

-Correu: String

-Tel: Int

-Vip: Boolean

+Reserva crearReserva()

+void apuntarFalta()

+String getInforme(Client)

}

```
class Gerent {
-Identificador: Int
}
class EspecifiacioMoto {
-Marca:String
-Model:String
}
Motorent "1"---"1" Vista: es comunica
Local "1"--- "lst_moto 0..*" Moto: disposa
Moto "1"---"1 estat" Estat: es troba
Moto "*"---"1 esp" EspecifiacioMoto : te
Motorent "1"--"lst_local *" Local: te
Motorent "1"---"lst_usuari *" Usuari: disposa
Local "1"---"lst_reserva *" Reserva: emmagatzema
Reserva "1"---"dataR 1" Data: es va fer
Reserva "1"---"dataD 1" Data: devolucio
Reserva "1"---"dataRe 1" Data: recollida
Gerent "1" --- "lo 1" Local: gestiona
Client "1" --- "adr 1" Adreca: viu
Client "1" --- "cc 1" CompteCorrent: te
Client "1" --- "lst_hist 0..*" Reserva: historial
Reserva "1"---"m 1" Moto: te
Local "1"---"adr 1" Adreca: es troba
Reserva "1" --- "origen 1" Local: origen
Reserva "1" --- "desti 1" Local: desti
Client "1" --- "dataN 1" Data: va neixer
Client "1" --- "dataIns 1" Data: va inscriure
@enduml
```