L2 – MotoRent

Entrega de la pràctica – MotoRent

Autors:

Joaquim Salvadó
NIUB: 16509920

Astor Prieto
NIUB: 16445586

Carles Toujouse
NIUB: 16478685

Índex

Breu descripció de la pràctica

MotoRentals, una empresa dedicada al lloguer de motos, ha demanat que desenvolupem el seu servei web anomenat MotoRent. En aquesta segona entrega d'aquesta pràctica, presentem els diagrames de seqüencia de 8 cassos d'ús i millorem tota la primera entrega en base al que ens feia falta en aquesta segona entrega, incloent cassos d'ús nous i un model de domini nou.

Llista de cassos d'ús

Actor: Gerent

- UC15: Donar alta VIP
- UC12: Veure locals amb menys de 5 motos
- UC13: Veure locals amb més del 75% de motos
- UC14: Gestionar motos local
- UC9: Entregar moto
- UC10: Veure informe
- UC11: Veure motos d'un local
- UC8: Finalitzar reserva

Actor: Sistema

- UC16: Generar informe
- UC15: Donar alta VIP

Actor: Client

- UC7: Veure motos disponibles d'un local
- UC4: Donar-se de baixa
- UC5: Modificar la reserva
- UC6: Fer log-out
- UC3: Fer reserva

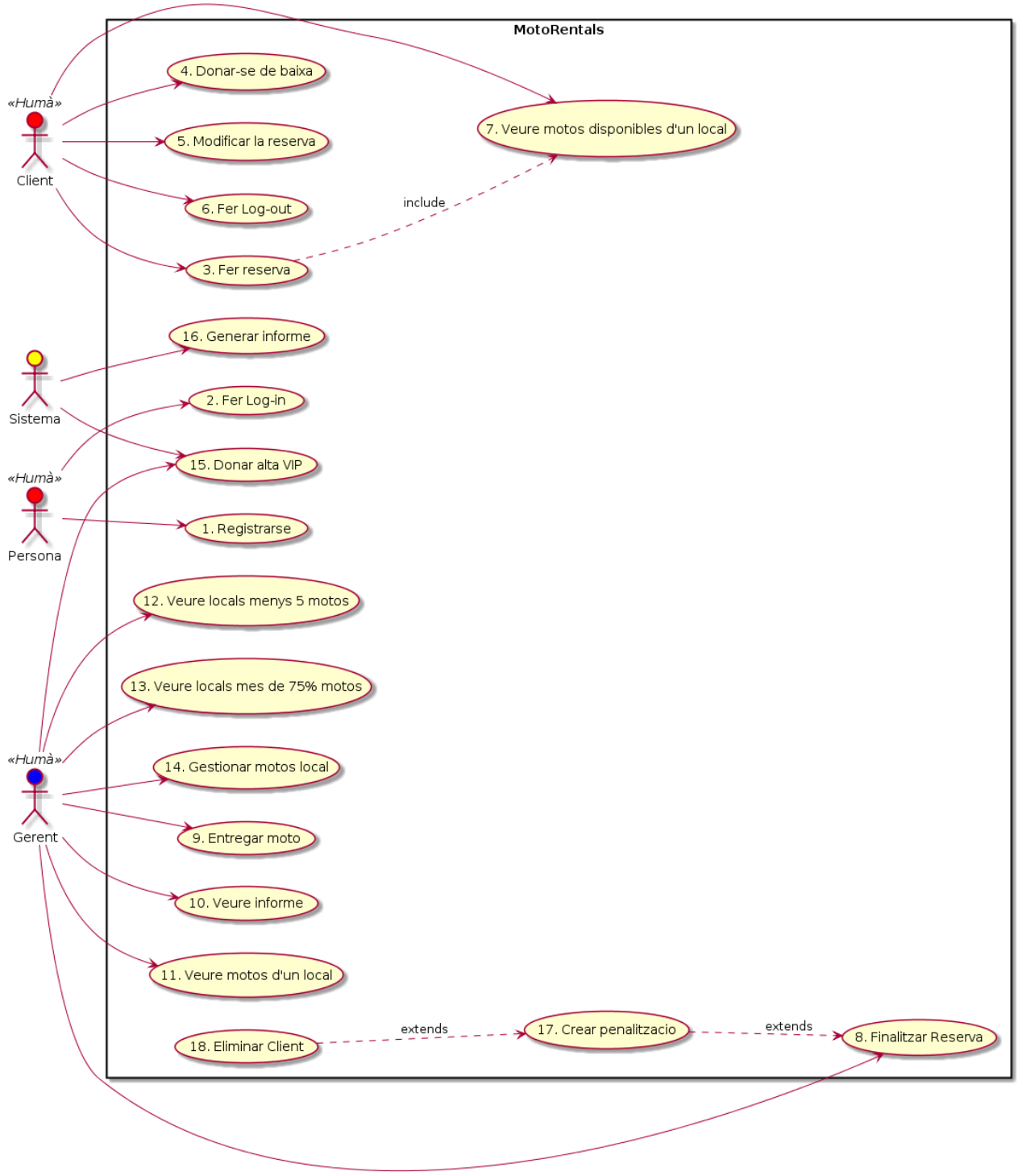
Actor: Persona

- UC1: Registrar-se
- UC2: Fer log-in

Diagrama de cassos d'ús

A continuació es mostra el codi plantUML del diagrama de cassos d'us:

```
@startuml
left to right direction
skinparam packageStyle rect
actor Client as user <<Humà>> #red
actor Persona as nologuser <<Humà>> #red
actor Gerent as owner <<Humà>> #blue
actor Sistema as time #yellow
rectangle MotoRentals{
(1. Registrarse) as UC1
(2. Fer Log-in) as UC2
(3. Fer reserva) as UC3
(4. Donar-se de baixa) as UC4
(5. Modificar la reserva) as UC5
(6. Fer Log-out) as UC6
(7. Veure motos disponibles d'un local) as UC7
(8. Finalitzar Reserva) as UC8
(9. Entregar moto) as UC9
(10. Veure informe) as UC10
(11. Veure motos d'un local) as UC11
(12. Veure locals menys 5 motos) as UC12
(13. Veure locals mes de 75% motos) as UC13
(14. Gestionar motos local) as UC14
(15. Donar alta VIP) as UC15
(16. Generar informe) as UC16
(17. Crear penalitzacio) as UC17
(18. Eliminar Client) as UC18
(UC17) ..> (UC8) : extends
(UC18) ..> (UC17) : extends
(UC3) ..> (UC7) : include
user --> (UC3)
user --> (UC4)
user --> (UC5)
user --> (UC6)
user --> (UC7)
owner --> (UC8)
owner --> (UC9)
owner --> (UC10)
owner --> (UC11)
owner --> (UC12)
owner --> (UC13)
owner --> (UC14)
owner --> (UC15)
time --> (UC15)
time --> (UC16)
nologuser --> (UC1)
nologuser --> (UC2)
}
@enduml
```


MotoRentals
«Humà»
Client
4. Donar-se de baixa
5. Modificar la reserva
6. Fer Log-out
3. Fer reserva
7. Veure motos disponibles d'un local
include
Sistema
16. Generar informe
2. Fer Log-in
15. Donar alta VIP
«Humà»
Persona
1. Registrarse
«Humà»
Gerent
12. Veure locals menys 5 motos
13. Veure locals mes de 75% motos
14. Gestionar motos local
9. Entregar moto
10. Veure informe
11. Veure motos d'un local
18. Eliminar Client
extends
17. Crear penalitzacio
extends
8. Finalitzar Reserva

Cassos d’ús textuals

A continuació es troba tots el cassos d’ús textuals.

	UC1 – Registrar-se
Descripció	La persona es registra al sistema
Actors	Persona
Precondicions	No estar logat
Flux bàsic	1- La persona selecciona l'opció de registrar-se 2- El sistema demana el nom d'usuari 3- La persona introdueix un nom d'usuari, pren enter, el sistema ho llegeix i comprova la disponibilitat del nom i guarda el nom 4- El sistema demana la contrasenya 5- La persona introdueix la contrasenya 6- El sistema llegeix la contrasenya i mostra el missatge «Registre completat» 7- El sistema mostra el menú principal
Flux alternatiu	3.1- Si el nom d'usuari ja existeix el sistema en demana un de nou
Post condicions	La persona queda registrada

	UC2 – Fer Log-in
Descripció	La persona inicia la sessió
Actors	Persona
Precondicions	No estar logat
Flux bàsic	1- La persona selecciona l'opció de fer log-in 2- El sistema demana introduir usuari i contrasenya 3- L'usuari introdueix el seu usuari i la seva contrasenya 4- Si l'usuari es client, mostra el menú de client
Flux alternatiu	4.1- Si la persona es admin, mostra el menú d'admin 4.2- Si les dades introduïdes no corresponen amb cap usuari ni contrasenya registrat, demana tornar a introduir les dades
Post condicions	El sistema reconeix a la persona, ja sigui client o admin

	UC3 – Fer reserva
Descripció	El client reserva una moto
Actors	Client i gerent
Precondicions	Estar logat com a client
Flux bàsic	1- El client entra a l'opció de reservar una moto 2- El client indica el local on recollirà la moto 3- La consola ensenya al client totes les motos que hi ha per llogar 4- El client indica la moto que vol reservar 5- El client indica el local on deixarà la moto 6- El client indica la data de recollida de la moto 7- El client indica la data de retorn de la moto 8- El sistema recull les dades i retorna un codi únic per al client
Flux alternatiu	3.1- Si la moto no esta disponible, fa escollir una de diferent al client 7.1- Si la data de retorn és abans de la data de recollida, dóna missatge d'error
Post condicions	El client te el codi de reserva

	UC4 – Donar-se de baixa
Descripció	El client es dóna de baixa del sistema
Actors	Client
Precondicions	Estar logat com a client
Flux bàsic	1- El client indica que es vol donar de baixa 2- El sistema demana una confirmació en forma de password.
Flux alternatiu	3.1- Si el client falla la confirmació el sistema avorta la baixa i demana que s'intenti més tard
Post condicions	El client deixa de estar logat i perd el seu compte

	UC5 – Modificar reserva
Descripció	El client modifica la seva reserva
Actors	Client
Precondicions	Que el client tingui feta una reserva i estigui logat
Flux bàsic	1- El client fa logg-in 2- Escull la opció per a modificar una reserva. 3- Canvia els paràmetres de la reserva i el sistema els guarda de nou
Flux alternatiu	3.1- El client pot escollir si modifica la hora de tornada de la reserva o el local de destí
Post condicions	El client ha modificat el local de destí o la hora de la seva reserva

	UC6 – Fer Log-out
Descripció	El client o gerent surten de la seva sessió
Actors	Client i gerent
Precondicions	Estar logat com a client o gerent
Flux bàsic	1- El client o gerent escull la opció de fer log-out 2- El sistema mostra el menú de persona
Flux alternatiu	
Post condicions	El client o gerent estan fora del sistema

	UC7 – Veure motos disponibles d'un local
Descripció	El client veu totes les motos disponibles d'un local
Actors	Client
Precondicions	Estar logat com a client
Flux bàsic	1- El client selecciona l'opció de veure les motos disponibles d'un local 2- El sistema mostra una llista amb tots els locals i pregunta de quin local vol veure les motos 3- El client selecciona un local 4- El sistema mostra per pantalla totes les motos disponibles del local
Flux alternatiu	1.1- Durant el procés de reserva el sistema mostra automàticament totes les motos disponibles del local en concret
Post condicions	L'usuari ha vist totes les motos que pot llogar

	UC8 – Finalitzar reserva
Descripció	Al gerent li arriba una moto
Actors	Gerent
Precondicions	Que un client hagi llogat la moto i estar logat com a gerent
Flux bàsic	1- El gerent rep una moto llogada amb el codi de la reserva d'aquesta 2- El gerent selecciona la opció de finalitzar reserva 3- El gerent comprova el codi i si tot està en ordre, finalitza la reserva
Flux alternatiu	2.1- Si la moto te algun desperfecte, el gerent crea una penalització al usuari que ha llogat la moto 2.1.1- Si aquest usuari te 3 penalitzacions acumulades, el gerent eliminarà el client
Post condicions	La moto queda retornada al local corresponent

	UC9 – Entregar moto
Descripció	El gerent entrega una moto a un client
Actors	Gerent
Precondicions	El client ha de tenir un codi únic i estar logat com a gerent
Flux bàsic	1- El gerent rep un codi únic de reserva 2- El gerent selecciona la opció de entregar moto 3- El gerent comprova el codi al sistema 4- El gerent entrega la moto corresponent al client
Flux alternatiu	
Post condicions	El gerent ha entregat la moto

	UC10 – Veure informe
Descripció	El gerent veu l'informe
Actors	Gerent
Precondicions	Estar logat com a gerent
Flux bàsic	1- El gerent selecciona l'opció de veure l'informe 2- El sistema mostra el informe per pantalla
Flux alternatiu	
Post condicions	El gerent ha vist el informe

	UC11 – Veure motos d'un local
Descripció	El gerent veu totes les motos d'un local en concret
Actors	Gerent
Precondicions	Estar logat com a gerent
Flux bàsic	1- El gerent selecciona l'opció de veure totes les motos d'un local 2- El sistema li ensenya tots els locals 3- El gerent selecciona un local en concret 4- El sistema li ensenya totes les motos que hi ha en aquell local
Flux alternatiu	
Post condicions	El gerent ha vist totes les motos que hi ha un local

	UC12 – Veure locals amb menys de 5 motos
Descripció	El gerent veu tots els locals amb menys de 5 motos
Actors	Gerent
Precondicions	Estar logat com a gerent
Flux bàsic	1- El gerent selecciona l'opció de veure tots els locals amb menys de 5 motos 2- El sistema mostra per pantalla una llista amb tots els locals amb menys de 5 motos
Flux alternatiu	2.1- Si no hi ha cap local amb menys de 5 motos, el sistema ho indica així per pantalla
Post condicions	El gerent ha vist tots els locals amb menys de 5 motos

	UC13 – Veure locals amb més d'un 75% de motos
Descripció	El gerent veu tots els locals amb més d'un 75% de motos
Actors	Gerent
Precondicions	Estar logat com a gerent
Flux bàsic	1- El gerent selecciona l'opció de veure tots els locals amb més d'un 75% de motos 2- El sistema mostra per pantalla una llista amb tots els locals amb més d'un 75% de motos
Flux alternatiu	2.1- Si no hi ha cap local amb més d'un 75% de capacitat, el sistema ho indica així per pantalla
Post condicions	El gerent ha vist tots els locals amb més d'un 75% de motos

	UC14 – Gestionar motos del local
Descripció	Al gerent se l'hi obre un menú on pot fer diferents coses
Actors	Gerent
Precondicions	Estar logat com a gerent
Flux bàsic	1- El gerent selecciona l'opció de gestionar les motos del local 2- El gerent selecciona un local amb menys de 5 motos 3- El gerent selecciona un local amb més d'un 75% de capacitat 4- El gerent indica el numero de motos que es transferiran 5- El sistema indica que la transferència de motos ja es pot efectuar
Flux alternatiu	
Post condicions	El gerent ha efectuat la transferència de motos

	UC15 – Donar alta VIP
Descripció	El sistema o el gerent concedeixen el VIP a un client
Actors	Sistema i gerent
Precondicions	Estar logat com a gerent
Flux bàsic	1- El gerent selecciona la opció de donar d'alta VIP a un client 2- El gerent rep una llista amb tots els clients 3- El gerent selecciona el client que vol fer VIP 4- El sistema fer VIP al client seleccionat
Flux alternatiu	1.1- El sistema dóna d'alta VIP a un client si aquest porta molt de temps essent un client de l'empresa
Post condicions	El client ara es VIP

	UC16 – Generar informe
Descripció	El sistema genera i envia un informe comercial
Actors	Sistema
Precondicions	Tenir algun client a qui poder fer el informe
Flux bàsic	1- El sistema recapta tota la informació dels clients automàticament i la recull en un sol arxiu per client 2- El sistema pot mostrar aquest arxiu en qualsevol moment
Flux alternatiu	
Post condicions	El informe de clients està creat

	UC17 – Crear penalització
Descripció	El gerent crea una penalització a un client
Actors	Gerent
Precondicions	El gerent ha hagut de rebre una moto en mal estat o el usuari ha arribat més tard de la hora establerta i estar logat com a gerent
Flux bàsic	1- El gerent selecciona la opció de crear penalització 2- El gerent penalitza al usuari amb una falta
Flux alternatiu	
Post condicions	El usuari queda penalitzat

	UC18 – Eliminar client
Descripció	El gerent elimina al usuari de l'empresa
Actors	Gerent
Precondicions	Estar logat com a gerent
Flux bàsic	1- El gerent selecciona la opció de eliminar un client 2- El sistema mostra la llista de client 3- El gerent en selecciona un i l'escriu per pantalla 4- El sistema elimina al client i ho notifica per pantalla
Flux alternatiu	1.1- El sistema elimina automàticament a un client amb més de 3 faltes
Post condicions	El usuari queda eliminat del sistema

Diagrama del model de domini

A continuació es mostra el codi plantUML del model de domini:

```
@startuml
class MotoRental {
}
class Persona {
 nom
 ..
 cognom
class Client {
 DNI
 ..
 telefon
 ..
 isVip
 ..
 conteBank
}
class Falta {
 cost
}
class Gerent {
 uniqueID
}
class Local{
 direccio
 ..
 tamany
}
class Vehicle {
 model
 ..
 color
 ..
 ID
 ..
 estat
}
class Reserva{
 temps
 ..
 codi
}
MotoRental "1" -- "*" Local : te
Local "1" -- "*" Vehicle : disposa
Client "1" -- "*" Falta : obte
Falta "*" -- "1" Gerent : crea
Client "1" -- "1" Reserva : solicita
MotoRental "1" -- "*" Client : te
Reserva "*" -- "1" Local : < origen
Reserva "*" -- "1" Local : > desti
Gerent "1" -- "1" Local : porta
Reserva "1" -- "0 .. 1" Falta : pot tenir
Vehicle "1" -- "1" Reserva : inclou
Persona <|-- Client
Persona <|-- Gerent
@endum
```

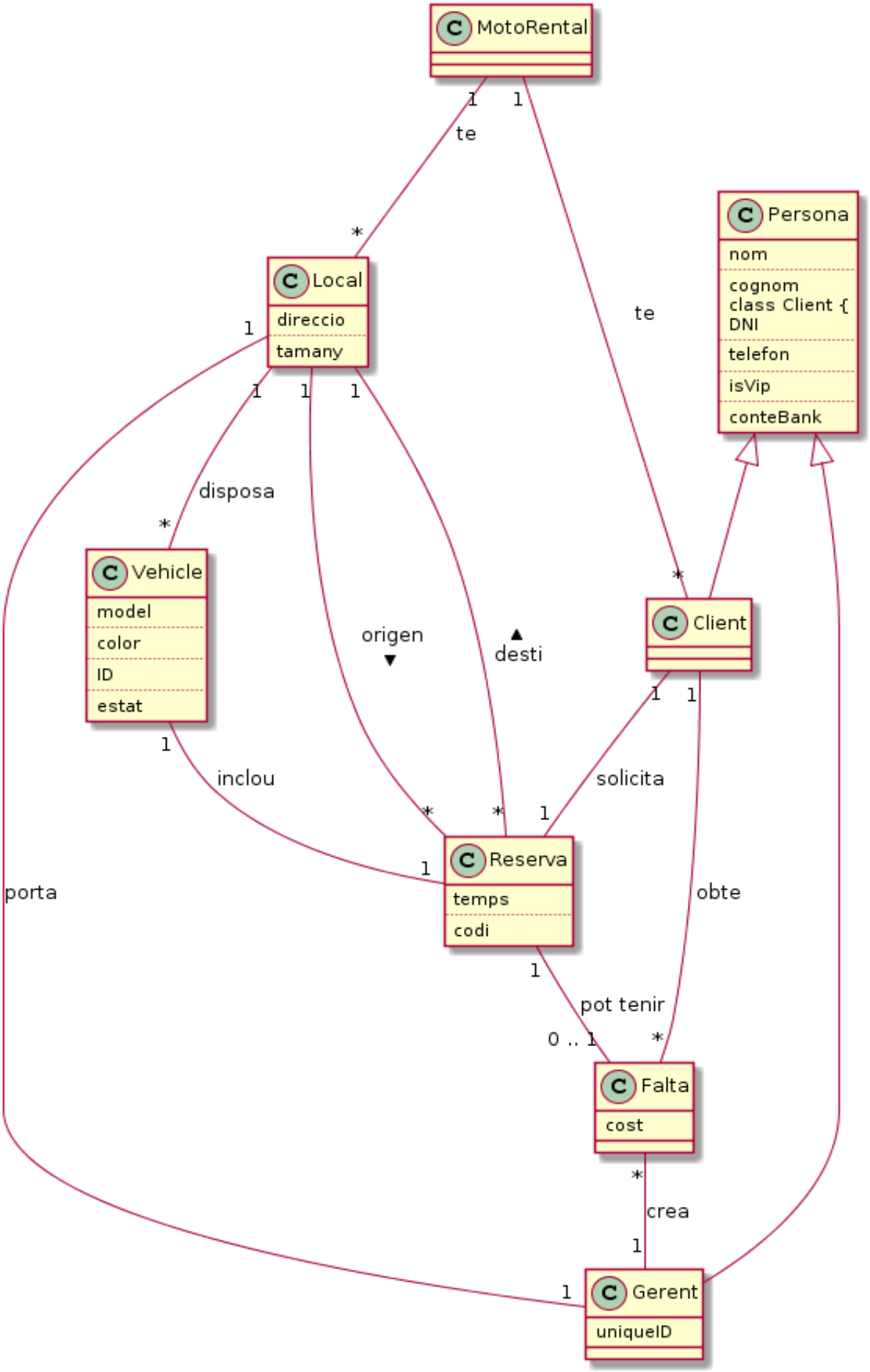

MotoRental
1
1
te
*
Local
direccio
tamany
1
1
1
1
disposa
*
Vehicle
model
color
ID
estat
1
inclou
1
origen
desti
*
*
Reserva
temps
codi
1
solicita
1
1
Client
te
*
Persona
nom
cognom
class Client {
DNI
telefon
isVip
conteBank
porta
1
obte
*
1
pot tenir
0 .. 1
*
Falta
cost
*
crea
1
1
Gerent
uniqueID

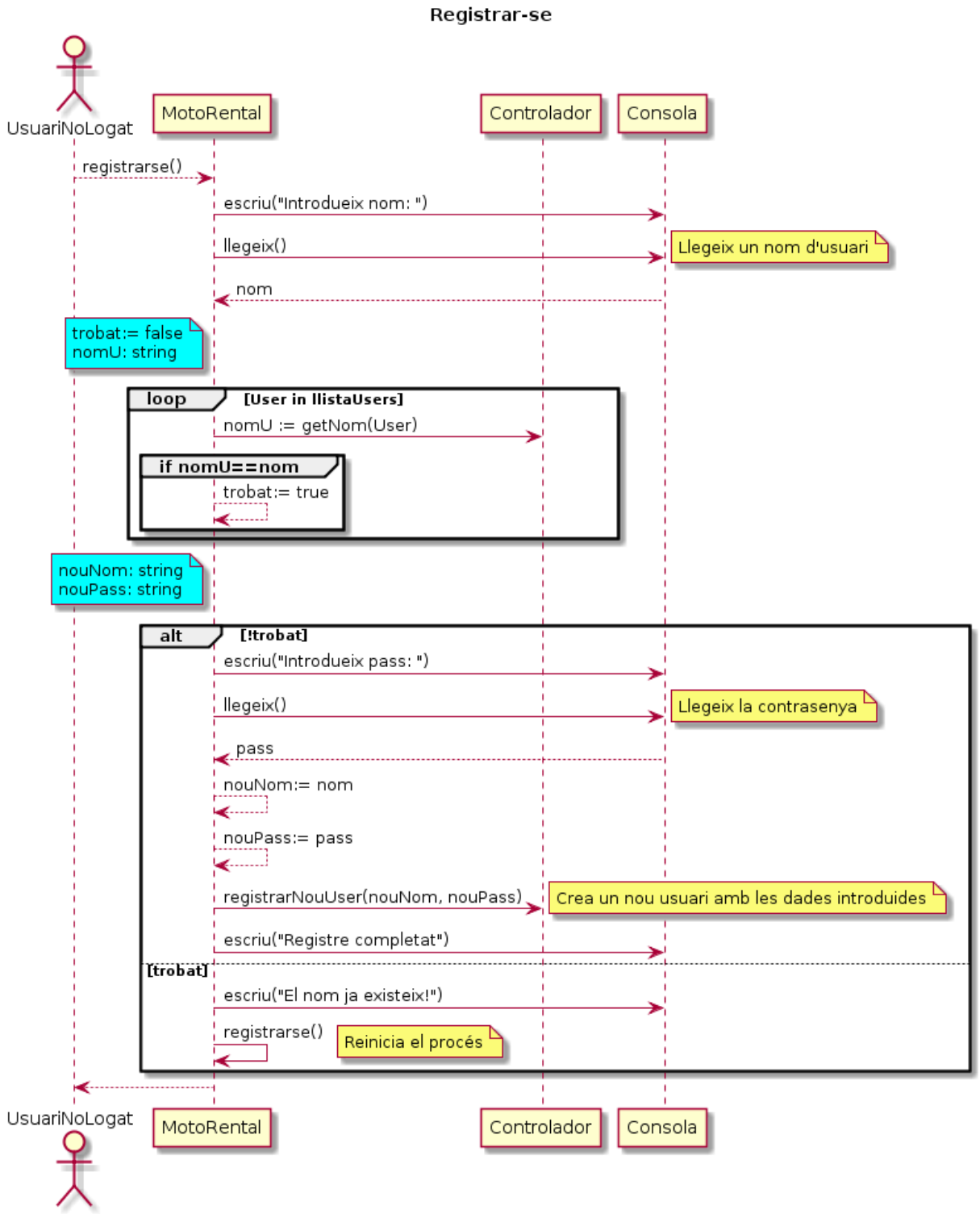
Diagrama de seqüencia de cassos d'us

1. Registrar-se

Codi del diagrama:

```
@startuml
title Registrar-se
actor UsuariNoLogat as user
participant MotoRental
participant Controlador
participant Consola
user --> MotoRental: registrarse()
MotoRental -> Consola: escriu("Introdueix nom: ")
MotoRental -> Consola: llegeix()
note right: Llegeix un nom d'usuari
Consola -->MotoRental: nom
note left of MotoRental #aqua
        trobat:= false
   nomU: string
end note
loop User in llistaUsers
          MotoRental -> Controlador: nomU := getNom(User)
          group if nomU==nom
             MotoRental -->MotoRental: trobat:= true
          end
       end
note left of MotoRental #aqua
          nouNom: string
     nouPass: string
end note
alt !trobat
   MotoRental -> Consola: escriu("Introdueix pass: ")
   MotoRental -> Consola: llegeix()
   note right: Llegeix la contrasenya
   Consola -->MotoRental: pass
   MotoRental --> MotoRental: nouNom:= nom
   MotoRental --> MotoRental: nouPass:= pass
   MotoRental -> Controlador: registrarNouUser(nouNom, nouPass)
   note right: Crea un nou usuari amb les dades introduides
   MotoRental -> Consola: escriu("Registre completat")
else trobat
   MotoRental -> Consola: escriu("El nom ja existeix!")
   MotoRental -> MotoRental: registrarse()
   note right: Reinicia el procés
end
MotoRental --> user
@enduml
```

Diagrama de seqüencia1

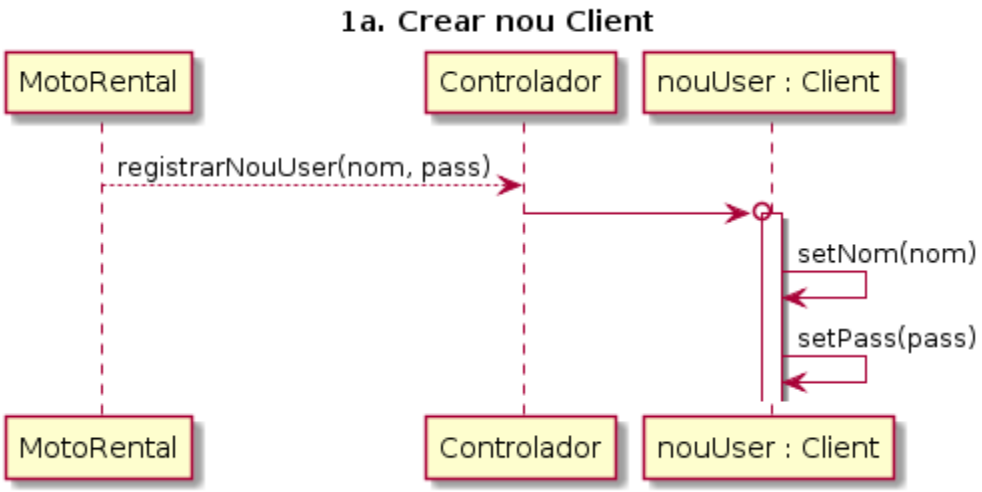


1a. Crear client

Codi del diagrama:

```
@startuml
title 1a. Crear nou Client
participant MotoRental
participant Controlador
participant "nouUser : Client" as Client
MotoRental --> Controlador: registrarNouUser(nom, pass)
Controlador ->o Client
activate Client
Client -> Client: setNom(nom)
Client -> Client: setPass(pass)
@enduml
```

Diagrama de seqüencia1a



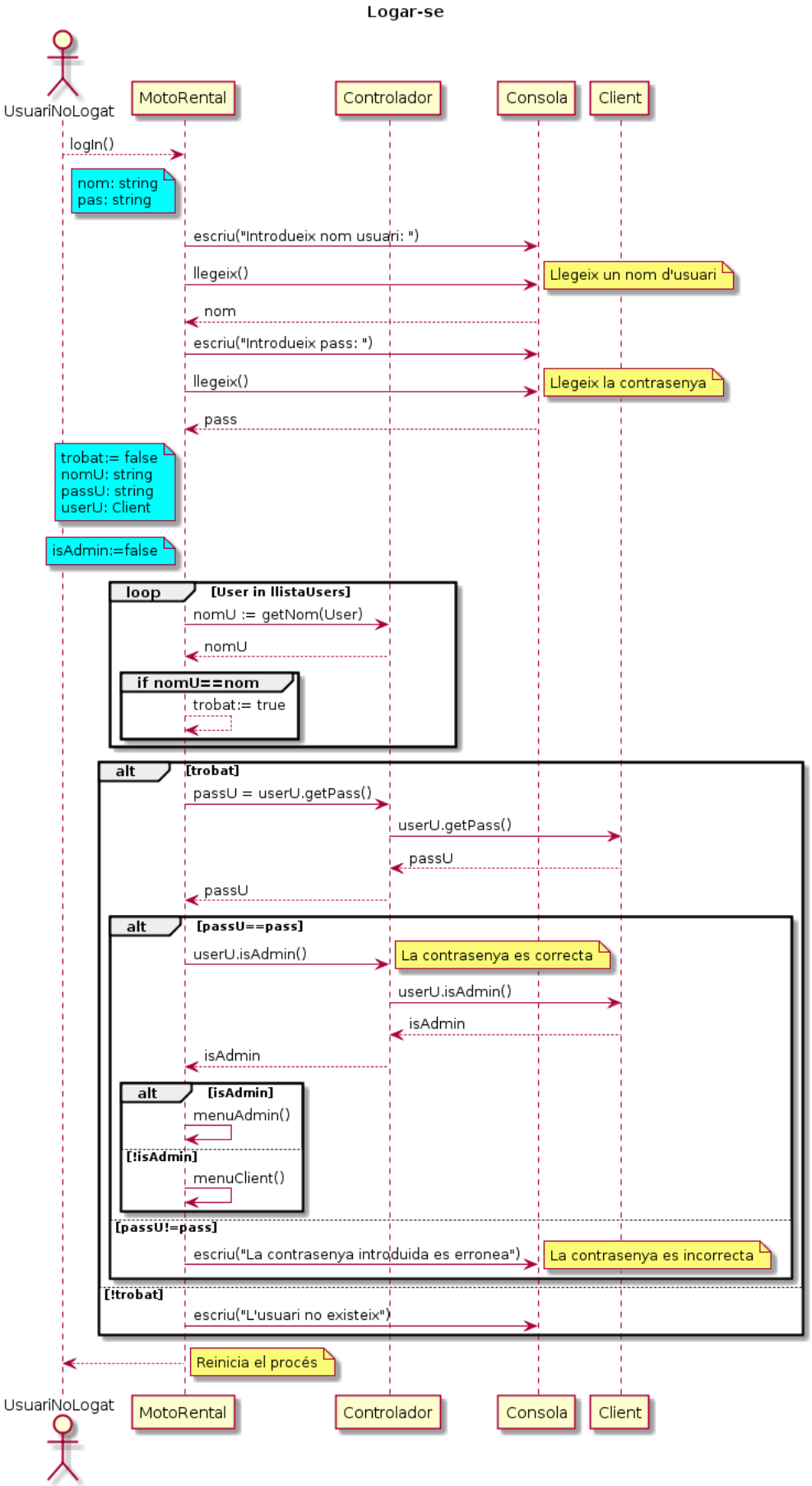
2. Logar-se

Codi del diagrama:

```
@startuml
title Logar-se
actor UsuariNoLogat as user
participant MotoRental
participant Controlador
participant Consola
participant Client
user --> MotoRental: logIn()
note left of MotoRental #aqua
        nom: string
    pas: string
end note
MotoRental -> Consola: escriu("Introdueix nom usuari: ")
MotoRental -> Consola: llegeix()
note right: Llegeix un nom d'usuari
Consola -->MotoRental: nom
MotoRental -> Consola: escriu("Introdueix pass: ")
    MotoRental -> Consola: llegeix()
    note right: Llegeix la contrasenya
    Consola -->MotoRental: pass
note left of MotoRental #aqua
    trobat:= false
    nomU: string
    passU: string
    userU: Client
    end note
note left of MotoRental #aqua
          isAdmin:=false
       end note
loop User in llistaUsers
          MotoRental -> Controlador: nomU := getNom(User)
          Controlador --> MotoRental : nomU
          group if nomU==nom
             MotoRental -->MotoRental: trobat:= true
          end
       end
alt trobat
    MotoRental -> Controlador: passU = userU.getPass()
    Controlador -> Client: userU.getPass()
    Client --> Controlador: passU
    Controlador--> MotoRental : passU
    alt passU==pass


       MotoRental -> Controlador: userU.isAdmin()
```

```
    note right: La contrasenya es correcta
    Controlador -> Client: userU.isAdmin()
    Client --> Controlador: isAdmin
    Controlador--> MotoRental : isAdmin

    alt isAdmin
      MotoRental -> MotoRental : menuAdmin()
    else !isAdmin
      MotoRental -> MotoRental : menuClient()
    end

  else passU!=pass

    MotoRental -> Consola: escriu("La contrasenya introduida es erronea")
    note right: La contrasenya es incorrecta
  end
else !trobat
  MotoRental -> Consola: escriu("L'usuari no existeix")
end
MotoRental --> user
note right: Reinicia el procés
@enduml
```

Diagrama de seqüencia2



3. Reservar moto

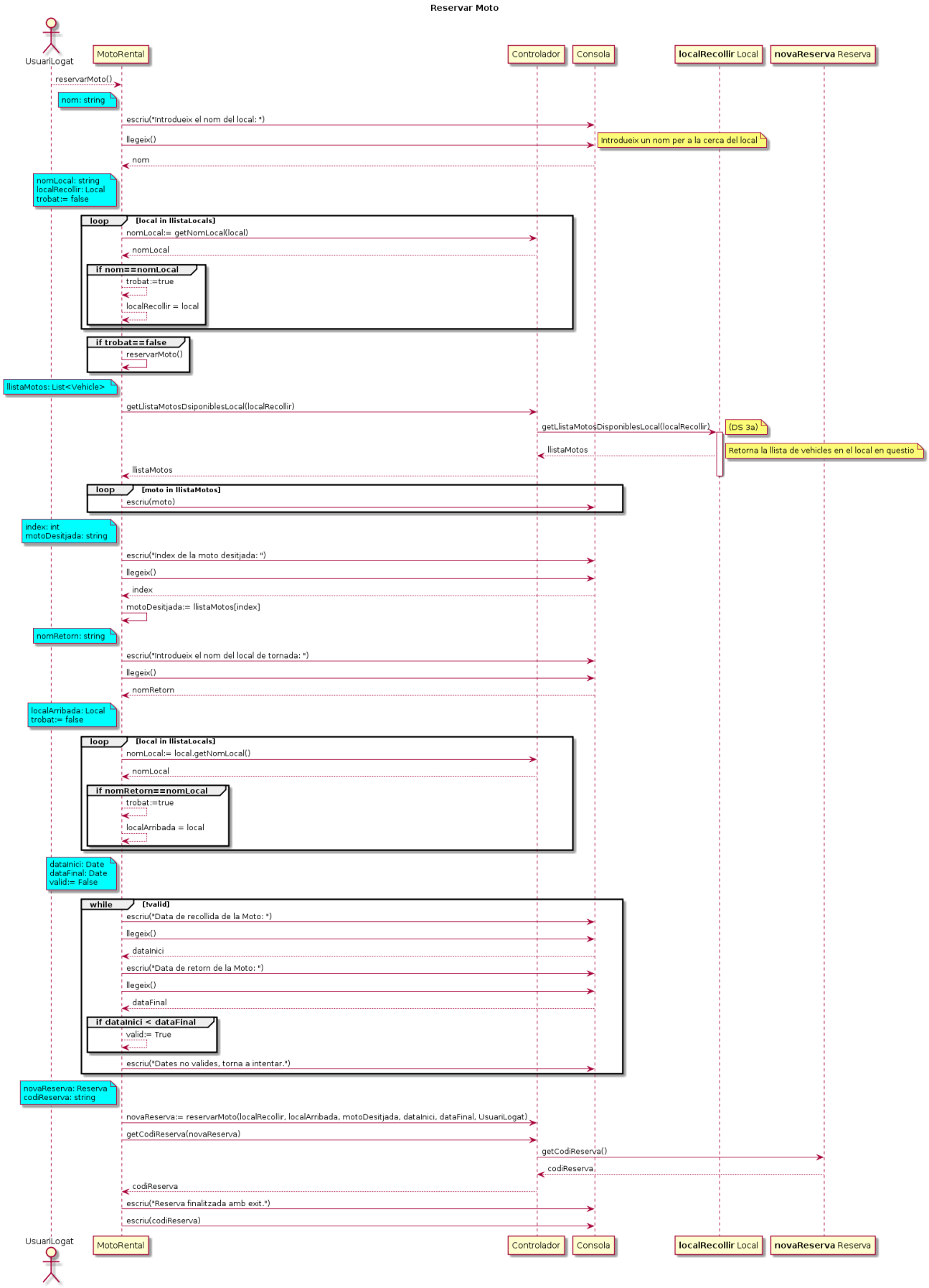
Codi del diagrama:

```
@startuml
title Reservar Moto
actor UsuariLogat as user
participant MotoRental
participant Controlador
participant Consola
participant "**localRecollir** Local" as Local
participant "**novaReserva** Reserva" as Reserva
user --> MotoRental: reservarMoto()
note left of MotoRental #aqua
   nom: string
end note
MotoRental -> Consola: escriu("Introdueix el nom del local: ")
MotoRental -> Consola: llegeix()
note right: Introdueix un nom per a la cerca del local
Consola --> MotoRental: nom
note left of MotoRental #aqua
   nomLocal: string
   localRecollir: Local
   trobat:= false
end note
loop local in llistaLocals
   MotoRental -> Controlador: nomLocal:= getNomLocal(local)
   Controlador --> MotoRental: nomLocal
   group if nom==nomLocal
      MotoRental --> MotoRental: trobat:=true
      MotoRental --> MotoRental: localRecollir = local
   end group
end loop
group if trobat==false
   MotoRental -> MotoRental : reservarMoto()
end group
note left of MotoRental #aqua
   llistaMotos: List<Vehicle>
end note
MotoRental -> Controlador : getLlistaMotosDsiponiblesLocal(localRecollir)
Controlador -> Local : getLlistaMotosDisponiblesLocal(localRecollir)
note right: (DS 3a)
activate Local
Local --> Controlador: llistaMotos
note right: Retorna la llista de vehicles en el local en questio
Controlador --> MotoRental : llistaMotos
```

```
deactivate Local
loop moto in llistaMotos

   MotoRental -> Consola : escriu(moto)
end loop
note left of MotoRental #aqua
   index: int
   motoDesitjada: string
end note
MotoRental -> Consola : escriu("Index de la moto desitjada: ")
MotoRental -> Consola : llegeix()
Consola --> MotoRental : index
MotoRental -> MotoRental : motoDesitjada:= llistaMotos[index]

note left of MotoRental #aqua
   nomRetorn: string
end note
MotoRental -> Consola : escriu("Introdueix el nom del local de tornada: ")
MotoRental -> Consola : llegeix()
Consola --> MotoRental : nomRetorn
note left of MotoRental #aqua
   localArribada: Local
   trobat:= false
end note
loop local in llistaLocals
   MotoRental -> Controlador: nomLocal:= local.getNomLocal()
   Controlador --> MotoRental: nomLocal
   group if nomRetorn==nomLocal
      MotoRental --> MotoRental: trobat:=true
      MotoRental --> MotoRental: localArribada = local
   end group
end loop
note left of MotoRental #aqua
   dataInici: Date
   dataFinal: Date
   valid:= False
end note
group while [!valid]
   MotoRental -> Consola : escriu("Data de recollida de la Moto: ")
   MotoRental -> Consola : llegeix()
   Consola --> MotoRental : dataInici
   MotoRental -> Consola : escriu("Data de retorn de la Moto: ")
   MotoRental -> Consola : llegeix()
   Consola --> MotoRental : dataFinal
   group if dataInici < dataFinal
      MotoRental --> MotoRental : valid:= True
   end group
   MotoRental -> Consola : escriu("Dates no valides, torna a intentar.")
```

```
end group
note left of MotoRental #aqua
   novaReserva: Reserva
   codiReserva: string
end note
MotoRental -> Controlador : novaReserva:= reservarMoto(localRecollir, localArribada,
motoDesitjada, dataInici, dataFinal, UsuariLogat)
MotoRental -> Controlador : getCodiReserva(novaReserva)
Controlador -> Reserva : getCodiReserva()
Reserva --> Controlador : codiReserva
Controlador --> MotoRental : codiReserva
MotoRental -> Consola : escriu("Reserva finalitzada amb exit.")
MotoRental -> Consola : escriu(codiReserva)
@enduml
```

Diagrama de seqüencia3

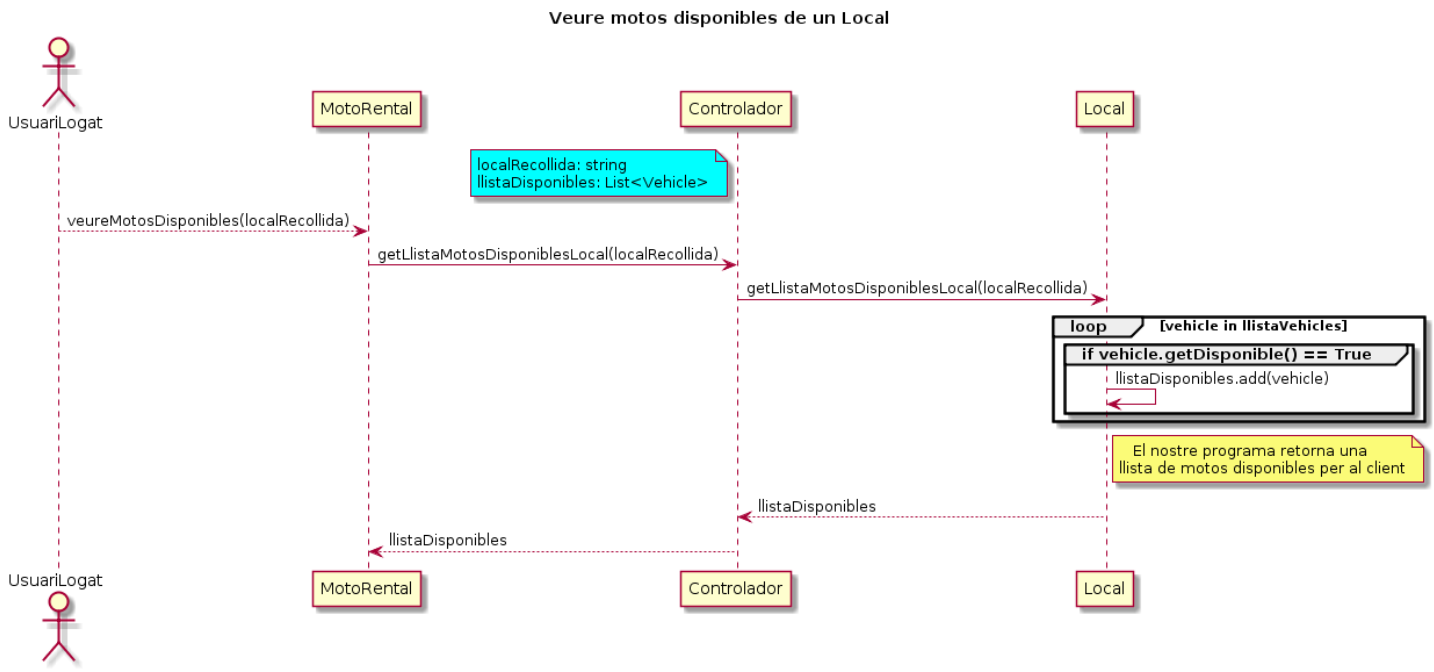


3a. Veure motos disponibles

Codi del diagrama:

```
@startuml
title Veure motos disponibles de un Local
actor UsuariLogat as user
participant MotoRental
participant Controlador
participant Local
note left of Controlador #aqua
      localRecollida: string
      llistaDisponibles: List<Vehicle>
   end note
user --> MotoRental: veureMotosDisponibles(localRecollida)
MotoRental -> Controlador: getLlistaMotosDisponiblesLocal(localRecollida)
Controlador -> Local: getLlistaMotosDisponiblesLocal(localRecollida)
loop vehicle in llistaVehicles
   group if vehicle.getDisponible() == True
      Local -> Local : llistaDisponibles.add(vehicle)
   end group
end loop
note right of Local
      El nostre programa retorna una
   llista de motos disponibles per al client
   end note
Local --> Controlador : llistaDisponibles
Controlador --> MotoRental : llistaDisponibles
@enduml
```

Diagrama de seqüencia3a

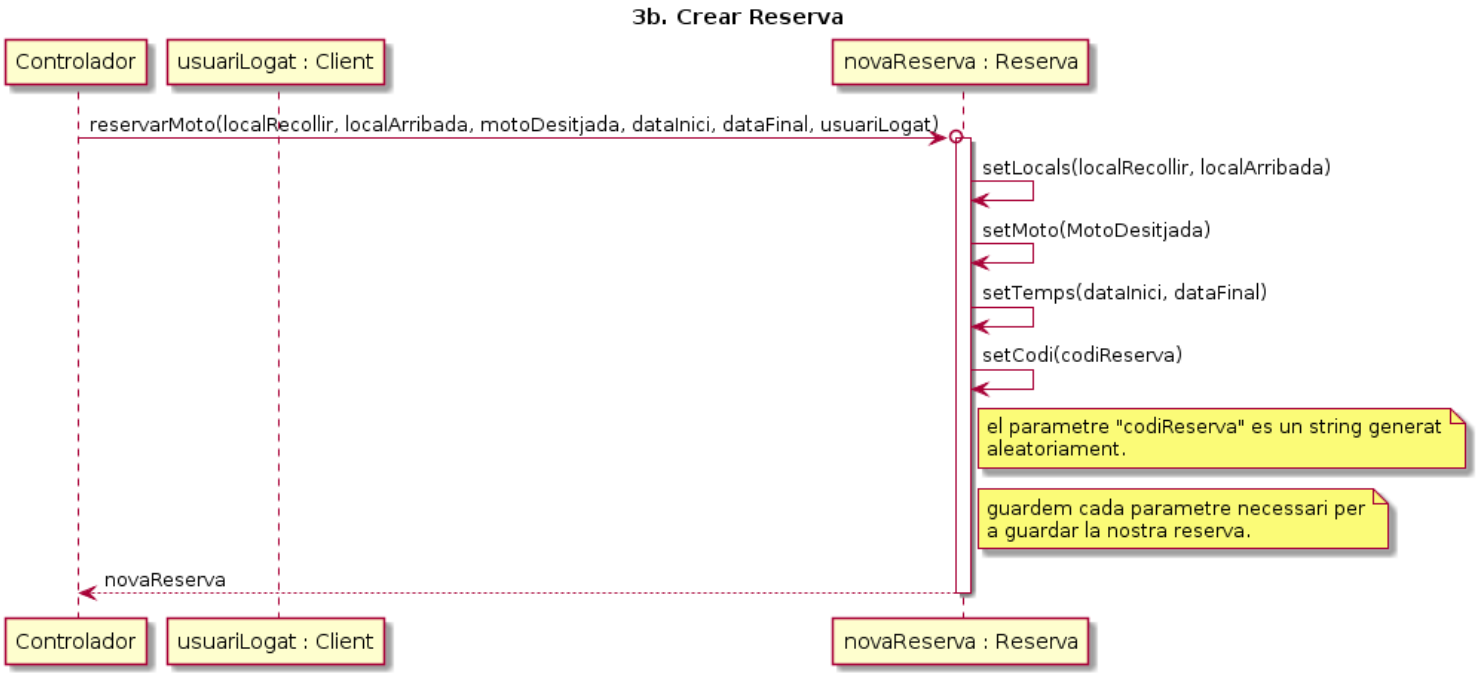


3b. Crear reserva

Codi del diagrama:

```
@startuml
title 3b. Crear Reserva
participant Controlador
participant "usuariLogat : Client" as Client
participant "novaReserva : Reserva" as Reserva
Controlador ->o Reserva: reservarMoto(localRecollir, localArribada, motoDesitjada,
dataInici, dataFinal, usuariLogat)
activate Reserva
Reserva -> Reserva: setLocals(localRecollir, localArribada)
Reserva -> Reserva: setMoto(MotoDesitjada)
Reserva -> Reserva: setTemps(dataInici, dataFinal)
Reserva -> Reserva: setCodi(codiReserva)
note right of Reserva
   el parametre "codiReserva" es un string generat
   aleatoriament.
end note
note right of Reserva
   guardem cada parametre necessari per
   a guardar la nostra reserva.
end note

Reserva --> Controlador : novaReserva
deactivate Reserva

@enduml
```

Diagrama de seqüencia3b



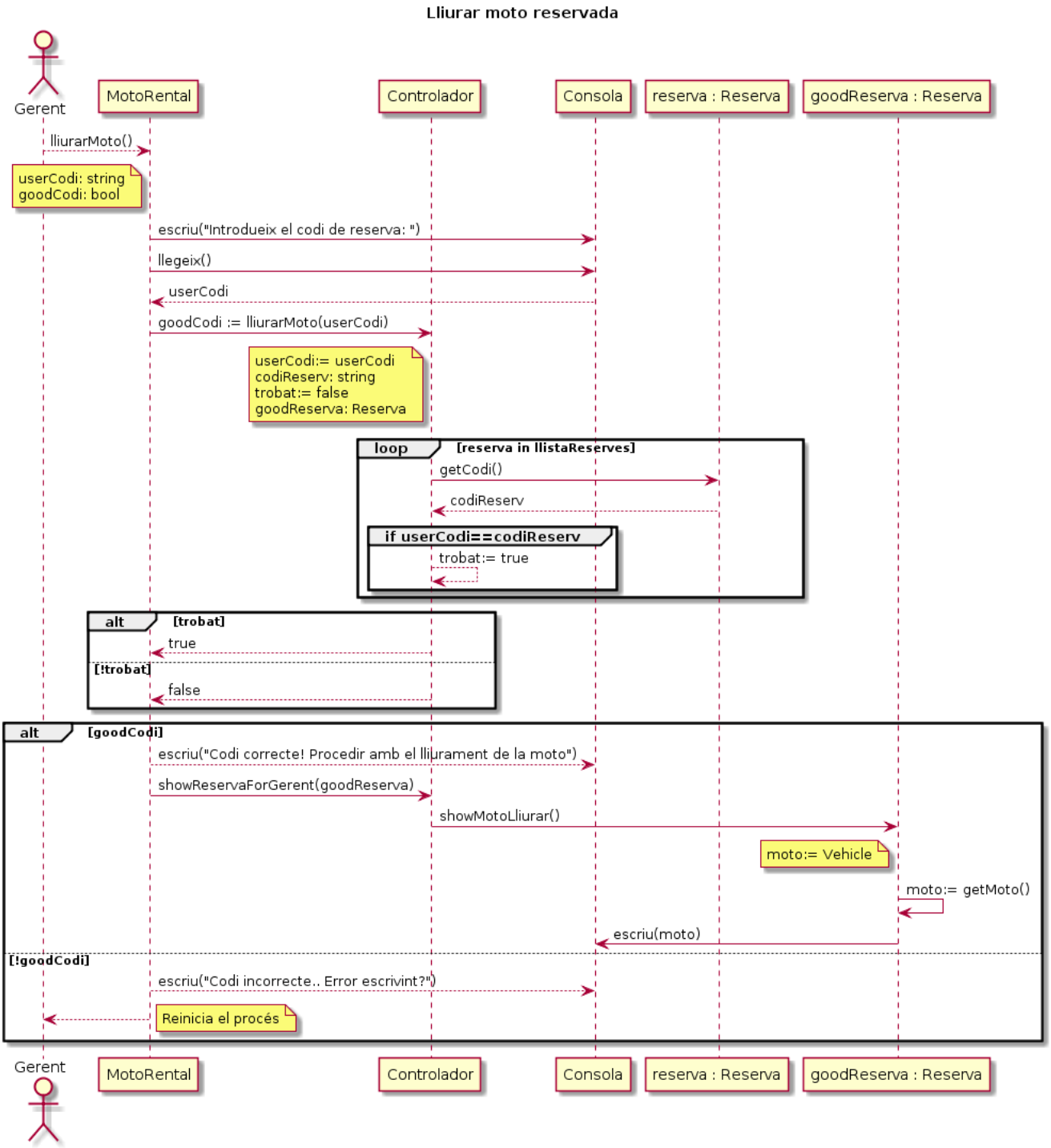
4. Lliurar moto

Codi del diagrama:

```
@startuml
title Lliurar moto reservada

actor Gerent as user

participant MotoRental
participant Controlador
participant Consola
participant "reserva : Reserva" as Reserva
participant "goodReserva : Reserva" as gReserva

user --> MotoRental: lliurarMoto()

note left of MotoRental
      userCodi: string
      goodCodi: bool
end note

MotoRental-> Consola: escriu("Introdueix el codi de reserva: ")
MotoRental-> Consola: llegeix()
Consola --> MotoRental: userCodi

MotoRental -> Controlador:  goodCodi := lliurarMoto(userCodi)

note left of Controlador
      userCodi:= userCodi
      codiReserv: string
      trobat:= false
      goodReserva: Reserva
end note
loop reserva in llistaReserves
   Controlador -> Reserva: getCodi()
   Reserva --> Controlador: codiReserv
   group if userCodi==codiReserv
      Controlador-->Controlador: trobat:= true
   end
end loop
alt trobat
   Controlador --> MotoRental: true
else !trobat
   Controlador --> MotoRental: false
end
alt goodCodi
   MotoRental--> Consola: escriu("Codi correcte! Procedir amb el lliurament de la moto")
   MotoRental-> Controlador: showReservaForGerent(goodReserva)
```

```
   Controlador -> gReserva: showMotoLliurar()
note left of gReserva
      moto:= Vehicle
end note
   gReserva -> gReserva: moto:= getMoto()
   gReserva -> Consola: escriu(moto)
else !goodCodi
   MotoRental--> Consola: escriu("Codi incorrecte.. Error escrivint?")
   MotoRental --> user
   note right: Reinicia el procés
end
@enduml
```

Diagrama de seqüencia4



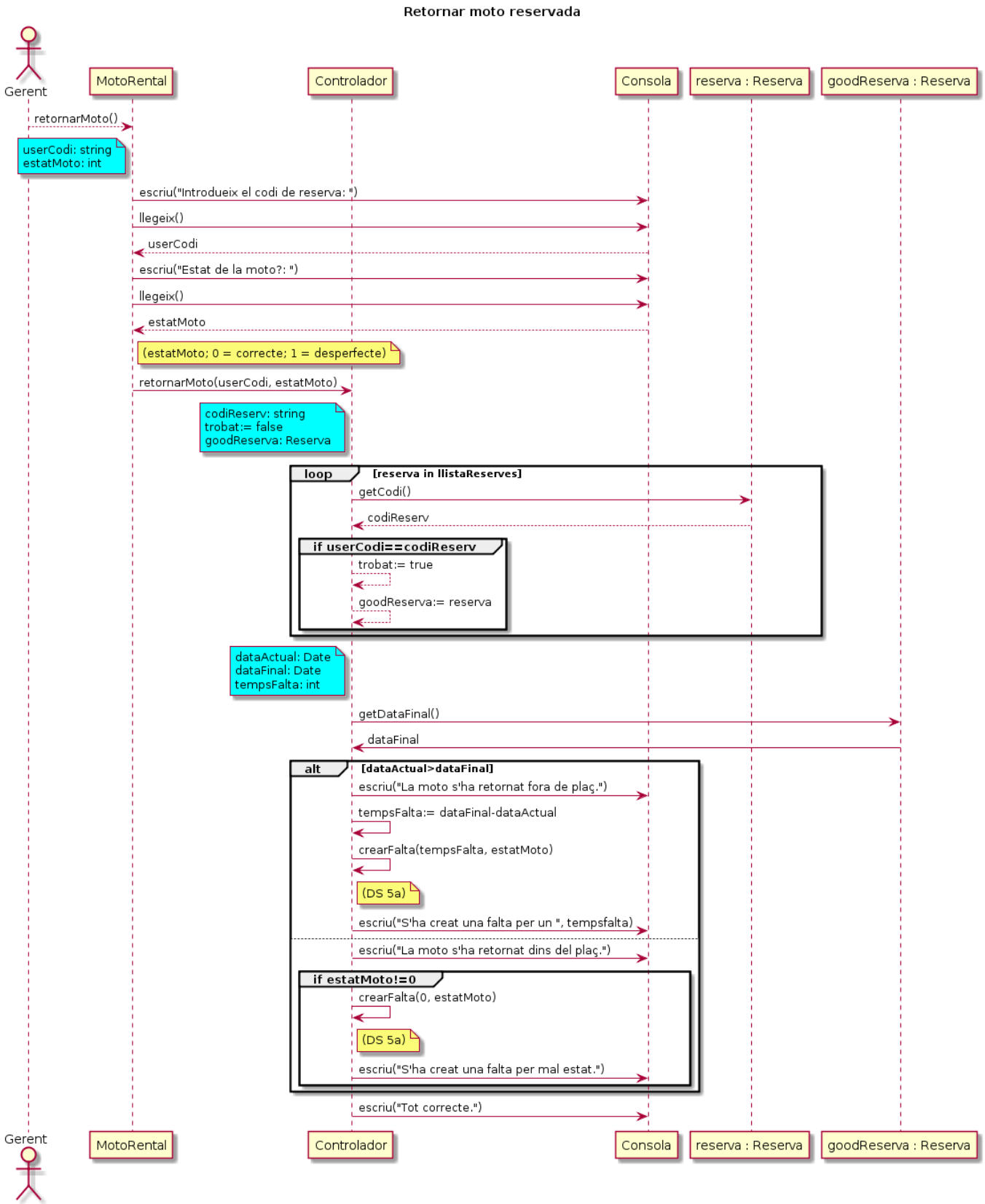
5. Retornar moto

Codi del diagrama:

```
@startuml
title Retornar moto reservada
actor Gerent as user
participant MotoRental
participant Controlador
participant Consola
participant "reserva : Reserva" as Reserva
participant "goodReserva : Reserva" as gReserva
user --> MotoRental: retornarMoto()
note left of MotoRental #aqua
      userCodi: string
      estatMoto: int
end note
MotoRental-> Consola: escriu("Introdueix el codi de reserva: ")
MotoRental-> Consola: llegeix()
Consola --> MotoRental: userCodi
MotoRental-> Consola: escriu("Estat de la moto?: ")
MotoRental-> Consola: llegeix()
Consola --> MotoRental: estatMoto
note right of MotoRental
      (estatMoto; 0 = correcte; 1 = desperfecte)
end note
MotoRental -> Controlador: retornarMoto(userCodi, estatMoto)
note left of Controlador  #aqua
      codiReserv: string
      trobat:= false
      goodReserva: Reserva
end note
loop reserva in llistaReserves
   Controlador -> Reserva: getCodi()
   Reserva --> Controlador: codiReserv
   group if userCodi==codiReserv
      Controlador-->Controlador: trobat:= true
      Controlador-->Controlador: goodReserva:= reserva
   end
end loop
note left of Controlador #aqua
      dataActual: Date
      dataFinal: Date
      tempsFalta: int
end note
Controlador -> gReserva: getDataFinal()
gReserva -> Controlador: dataFinal
alt dataActual>dataFinal
   Controlador -> Consola: escriu("La moto s'ha retornat fora de plaç.")
```

```
    Controlador -> Controlador: tempsFalta:= dataFinal-dataActual
    Controlador -> Controlador: crearFalta(tempsFalta, estatMoto)
    note right of Controlador
        (DS 5a)
    end note
    Controlador -> Consola: escriu("S'ha creat una falta per un ", tempsfalta)
else
    Controlador -> Consola: escriu("La moto s'ha retornat dins del plaç.")
    group if estatMoto!=0
        Controlador -> Controlador: crearFalta(0, estatMoto)
        note right of Controlador
        (DS 5a)
        end note
        Controlador -> Consola: escriu("S'ha creat una falta per mal estat.")
    end group
end
Controlador -> Consola: escriu("Tot correcte.")
@enduml
```

Diagrama de seqüencia5

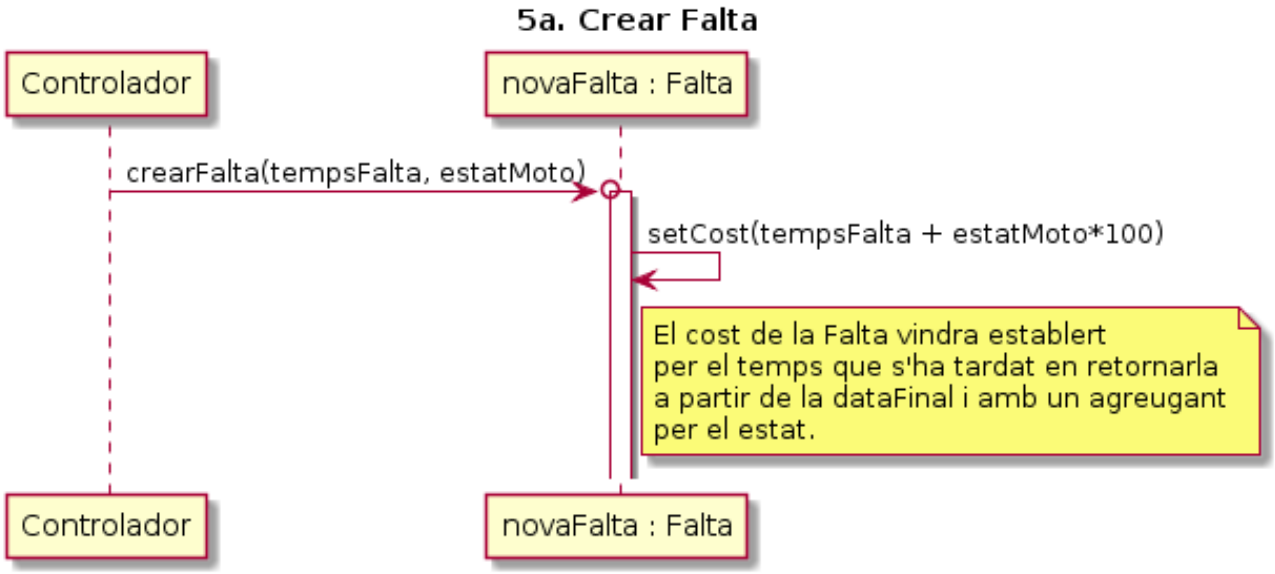


5a. Crear falta

Codi del diagrama:

```
@startuml
title 5a. Crear Falta
participant Controlador
participant "novaFalta : Falta" as Falta
Controlador ->o Falta: crearFalta(tempsFalta, estatMoto)
activate Falta
Falta -> Falta: setCost(tempsFalta + estatMoto*100)

note right of Falta
   El cost de la Falta vindra establert
   per el temps que s'ha tardat en retornarla
   a partir de la dataFinal i amb un agreugant
   per el estat.
end note
@enduml
```

Diagrama de seqüencia5a

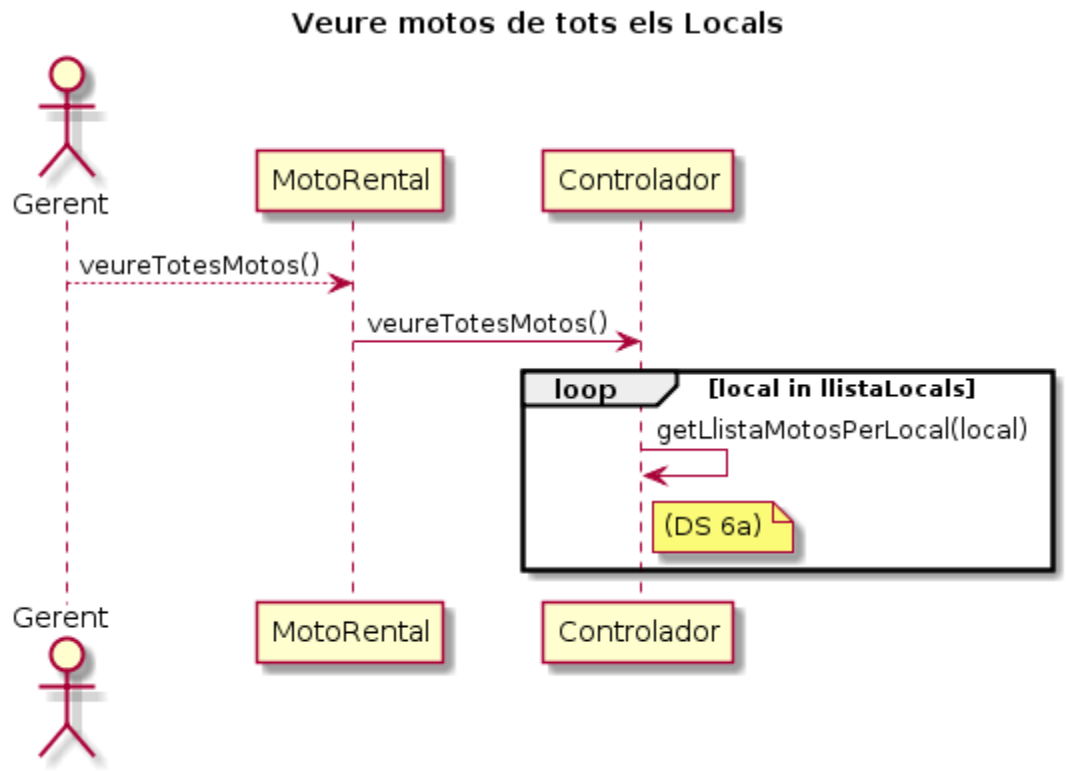


6. Motos de tots els locals

Codi del diagrama:

```
@startuml
title Veure motos de tots els Locals
actor Gerent as user
participant MotoRental
participant Controlador

user --> MotoRental: veureTotesMotos()
MotoRental -> Controlador: veureTotesMotos()
loop local in llistaLocals
   Controlador-> Controlador: getLlistaMotosPerLocal(local)
   note right of Controlador
     (DS 6a)
   end note

end loop
@enduml
```

Diagrama de seqüencia6

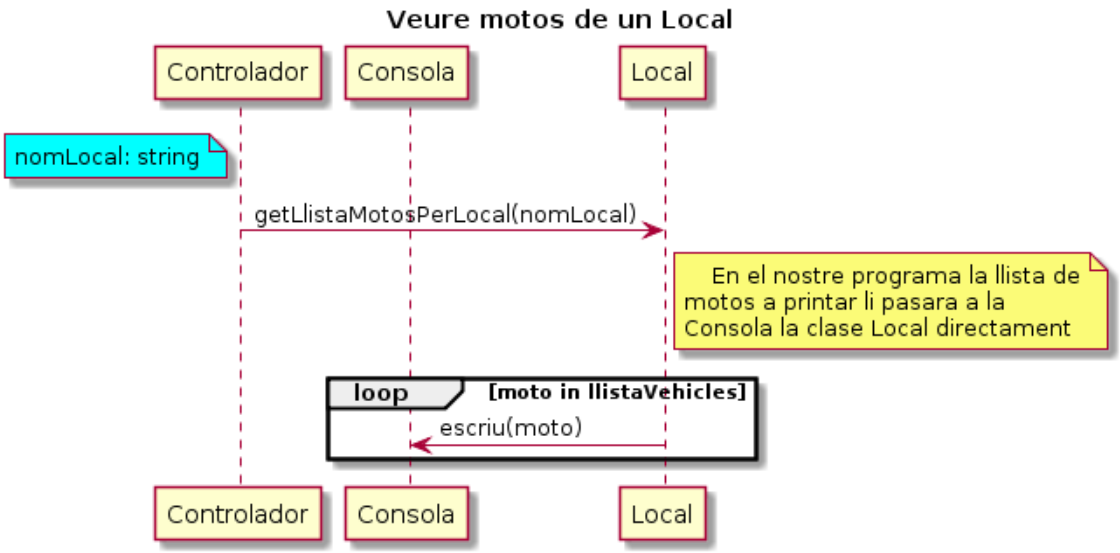


6a. Motos d'un local

Codi del diagrama:

```
@startuml
title Veure motos de un Local
participant Controlador
participant Consola
participant Local
note left of Controlador #aqua
      nomLocal: string
   end note
Controlador -> Local: getLlistaMotosPerLocal(nomLocal)
note right of Local
      En el nostre programa la llista de
   motos a printar li pasara a la
   Consola la clase Local directament
   end note
loop moto in llistaVehicles
   Local -> Consola: escriu(moto)
end loop
@enduml
```

Diagrama de seqüencia6a



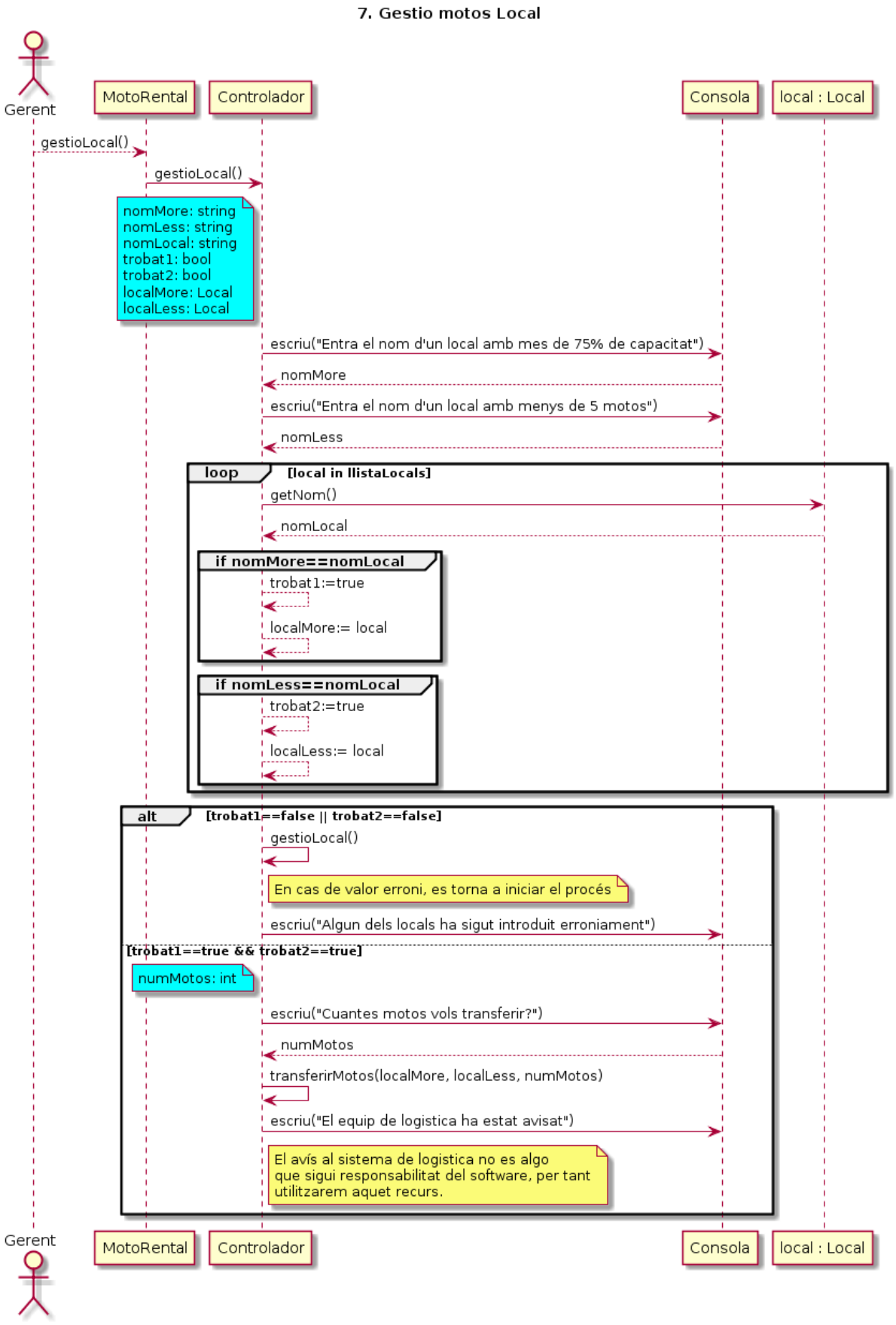
7. Gestionar local

Codi del diagrama:

```
@startuml
title 7. Gestio motos Local
actor Gerent as user
participant MotoRental
participant Controlador
participant Consola
participant "local : Local" as Local
user --> MotoRental: gestioLocal()
MotoRental -> Controlador: gestioLocal()
note left of Controlador #aqua
  nomMore: string
  nomLess: string
  nomLocal: string
  trobat1: bool
  trobat2: bool
  localMore: Local
  localLess: Local
end note
Controlador -> Consola: escriu("Entra el nom d'un local amb mes de 75% de capacitat")
Consola --> Controlador: nomMore
Controlador -> Consola: escriu("Entra el nom d'un local amb menys de 5 motos")
Consola --> Controlador: nomLess
loop local in llistaLocals
   Controlador -> Local: getNom()
   Local--> Controlador : nomLocal
   group if nomMore==nomLocal
      Controlador --> Controlador : trobat1:=true
      Controlador --> Controlador : localMore:= local
   end group
   group if nomLess==nomLocal
      Controlador --> Controlador : trobat2:=true
      Controlador --> Controlador : localLess:= local
   end group
end loop
alt trobat1==false || trobat2==false
   Controlador -> Controlador : gestioLocal()
   note right of Controlador: En cas de valor erroni, es torna a iniciar el procés
   Controlador -> Consola: escriu("Algun dels locals ha sigut introduit erroniament")
else trobat1==true && trobat2==true
   note left of Controlador #aqua
      numMotos: int
   end note
   Controlador -> Consola: escriu("Cuantes motos vols transferir?")
   Consola --> Controlador: numMotos
   Controlador -> Controlador: transferirMotos(localMore, localLess, numMotos)
```

```
    Controlador -> Consola: escriu("El equip de logistica ha estat avisat")
    note right of Controlador
        El avís al sistema de logistica no es algo
        que sigui responsabilitat del software, per tant
        utilitzarem aquet recurs.
    end note
end
@enduml
```

Diagrama de seqüencia7

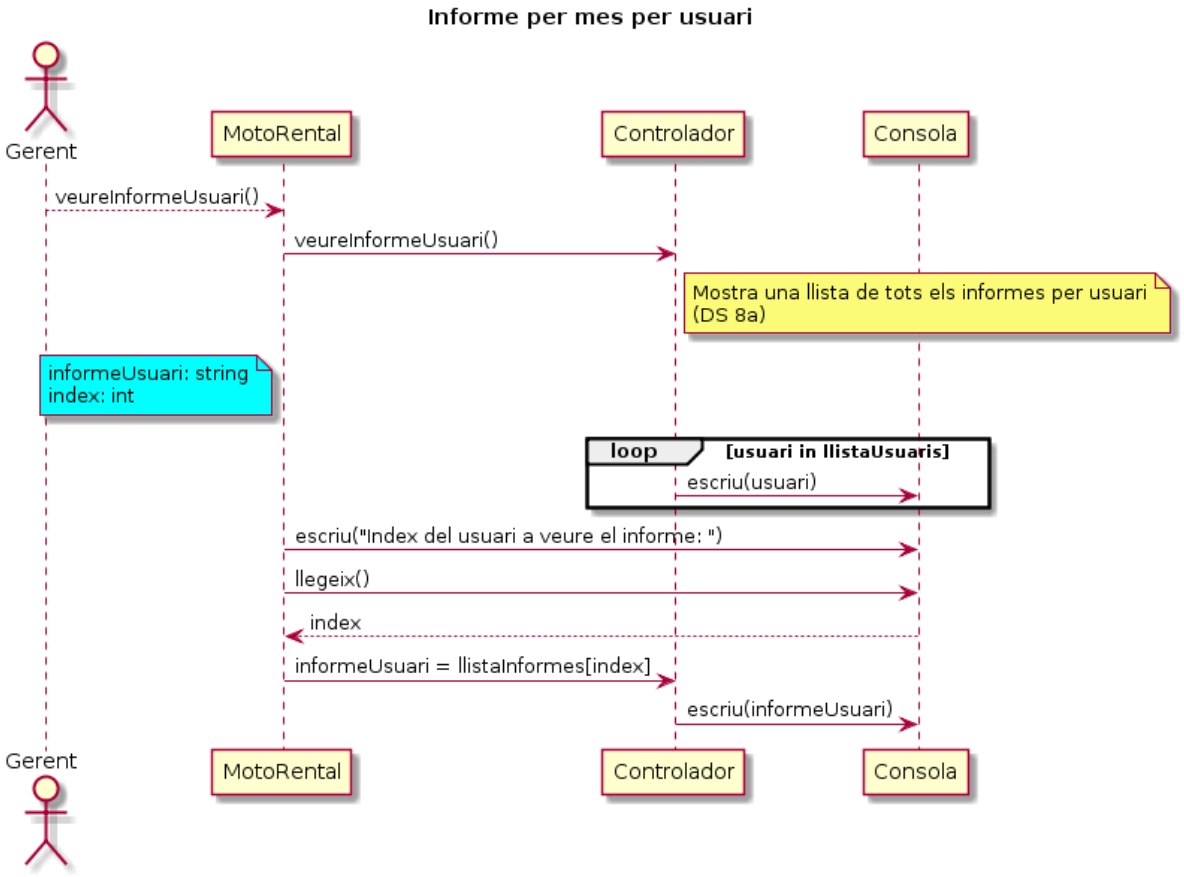


8. Veure informe

Codi del diagrama:

```
@startuml
title Informe per mes per usuari
actor Gerent as user
participant MotoRental
participant Controlador
participant Consola
user --> MotoRental: veureInformeUsuari()
MotoRental -> Controlador : veureInformeUsuari()
note right of Controlador
   Mostra una llista de tots els informes per usuari
   (DS 8a)
end note
note left of MotoRental #aqua
   informeUsuari: string
   index: int
end note
loop usuari in llistaUsuaris
   Controlador -> Consola : escriu(usuari)
end loop
MotoRental -> Consola : escriu("Index del usuari a veure el informe: ")
MotoRental -> Consola : llegeix()
Consola --> MotoRental : index
MotoRental -> Controlador : informeUsuari = llistaInformes[index]
Controlador -> Consola : escriu(informeUsuari)
@enduml
```

Diagrama de seqüencia8

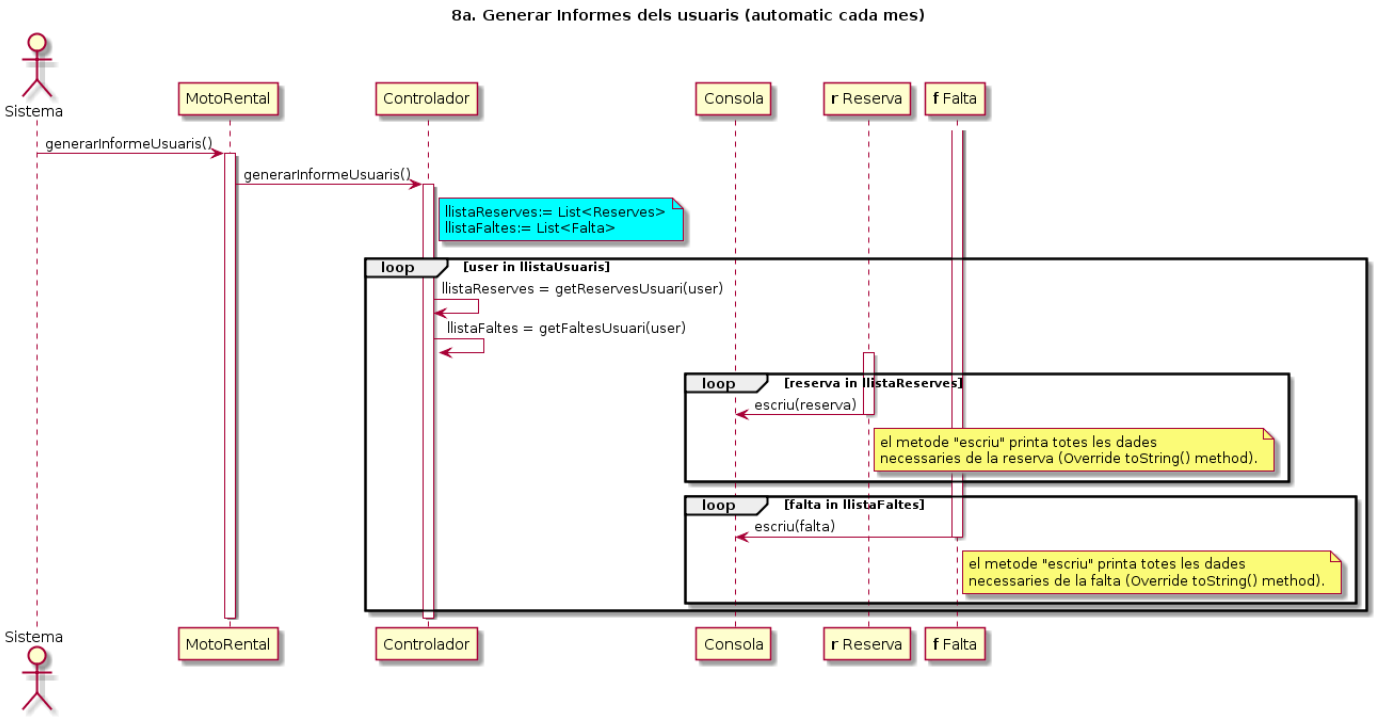


8a. Generar informe

Codi del diagrama:

```
@startuml
title 8a. Generar Informes dels usuaris (automatic cada mes)
actor Sistema as user
participant MotoRental
participant Controlador
participant Consola
participant "**r** Reserva" as Reserva
participant "**f** Falta" as Falta
user -> MotoRental : generarInformeUsuaris()
activate MotoRental
MotoRental -> Controlador : generarInformeUsuaris()
activate Controlador
note right of Controlador #aqua
    llistaReserves:= List<Reserves>
    llistaFaltes:= List<Falta>
end note
loop user in llistaUsuaris
    Controlador -> Controlador : llistaReserves = getReservesUsuari(user)
    Controlador -> Controlador : llistaFaltes = getFaltesUsuari(user)
    activate Reserva
    loop reserva in llistaReserves
        Reserva -> Consola : escriu(reserva)
        note right of Reserva
            el metode "escriu" printa totes les dades
            necessaries de la reserva (Override toString() method).
        end note
        deactivate Reserva
    end loop
    activate Falta
    loop falta in llistaFaltes
        Falta -> Consola : escriu(falta)
        note right of Falta
            el metode "escriu" printa totes les dades
            necessaries de la falta (Override toString() method).
        end note
        deactivate Falta
    end loop
end loop
deactivate Controlador
deactivate MotoRental
@enduml
```

Diagrama de seqüencia8a



Observacions

Al llarg d'aquesta segona entrega ens hem topat amb diversos problemes. Primer de tot hem hagut de modificar, crear i eliminar alguns cassos d'ús textuals ja que degut a errors de comprensió durant la primera entrega. Un cop arreglats aquests problemes també hem hagut de retocar codis i diagrames per tal de que tot quadres amb els nous cassos d'us. Al acabar de corregir tots els problemes de la pràctica 1 vam poder continuar amb la pràctica 2 on ens demanen fer el diagrama de seqüencia dels següents cassos d'us:

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
8. Informe al final de cada mes que ara fa l'administrador del sistema

Però en la nostre pràctica hi ha algun diagrama de seqüenciaque no es exactament com els que ens demanen fer, així que els diagrames de seqüenciaque nosaltres hem dissenyat són:

1. Registrar Usuari
   1a. Crear client
2. Logar-se
3. Reservar moto
   3a. Veure motos disponibles
   3b. Crear reserva
4. Lliurar moto
5. Retornar moto
   5a. Crear Falta
6. Veure motos de tots els locals
   6a. Veure motos d'un local
7. Gestionar local
8. Veure informe
   8a. Generar informe

Com es pot veure hem passat dels 8 diagrames originals a 14 de finals. Això es degut a que, en el diagrama de seqüencianumero 1, quan registra un client també ha de crear aquest client i per tal de fer-ho més entendible i clar ho hem dividit en 2 parts. El mateix passa en els altres diagrames de seqüenciaon hi ha més d'un diagrama.

Adjuntat en aquest fitxer es troba el projecte de NetBeans amb tot lo relacionat amb el controlador, model i vista del programa.

Distribució de la feina

Per tal de fer aquesta pràctica hem anat quedant durant diversos dies tots els membres del grup a la universitat per tal de comentar una mica entre tots cada apartat però a grans trets podem dir que l'Astor Prieto és va centrar en la programació i en crear diagrames de classes, en Joaquim Salvadó és va centrar en els cassos d'us textuals i correcció d'errors i en Carles Toujouse en crear el diagrama de domini i diagrames de seqüencia

Conclusions

Durant el transcurs d'aquesta pràctica em creat diferents diagrames de seqüencia. Dintre d'aquests diagrames hem hagut de crear molts mètodes i classes que posteriorment haurem de programar. Mentre creàvem els diagrames hem pogut veure tota la feina i complexitat amb la que ens trobarem a la hora de programar.