Gestire gli eventi

Bettarello, Mendozza - Progetto SAS Laboratorio 22/23

Informazioni generali

Nome caso d'uso: Gestire gli eventi
Portata: Sistema
Livello: Obiettivo Utente
Attore primario: Organizzatore
Parti Interessate: Chef, personale di servizio, cuoco
Pre-condizioni: L'attore deve essere autenticato come Organizzatore
Garanzie di successo o post-condizioni: L'evento è terminato

Scenario principale di successo

#	Attore	Sistema
1	Compila la scheda per l'evento	Creata la scheda per l'evento
2	Assegna uno Chef alla cucina	Chef assegnato all'evento
3	Effettua modifiche al menù	Modifiche al menù effettuate
Se desidera torna al passo 3 se no prosegue		
4	Assegna il personale ai turni	Personale di servizio assegnato ai turni
Se parte del personale non e' disponibile ritorna al passo 4 se no prosegue		
5	**Opzionalmente**, assegna un cuoco ad un determinato servizio	Cuoco assegnato per le preparazioni dell'ultimo minuto
6	**Opzionalmente**, definisce la ricorrenza dell'evento	Ricorrenza dell'evento definita
7	**Opzionalmente**, si aggiungono eventuali note e documentazione al termine dell'evento	Aggiunte la documentazione e le note riguardanti l'evento svoltosi
Termine del caso d'uso		

Estensione 1a

#	Attore	Sistema
1a.1	Apre una scheda evento pre-esistente	Aperta per eventuale modifica la scheda di un evento pre-esistente

Eccezione 1a.1a

#	Attore	Sistema
1a.1a.1	Apre una scheda evento pre-esistente	L'evento è in corso e la scheda non può essere aperta da nessuno
Termine del caso d'uso		

Estensione 1b

#	Attore	Sistema
1b.1	Elimina un evento esistente	Eliminato un evento esistente
Termine del caso d'uso		

Eccezione 1b.1a

#	Attore	Sistema
1b.1a.1	Elimina un evento esistente	L'evento è in corso e non può essere eliminato
Termine del caso d'uso		

Estensione 1c

#	Attore	Sistema
1c.1	Annulla un evento esistente	Annullato un evento esistente

Termine del caso d'uso

Eccezione 1c.1a

#	Attore	Sistema
1c.1a.1	Elimina un evento esistente	L'evento è in corso e non può essere annullato
Termine del caso d'uso		

Estensione 6a

#	Attore	Sistema
6a.1	Modifica la ricorrenza di un evento pre-esistente	Aperta per modifica la scheda di un evento ricorrente pre-esistente
Termine del caso d'uso		

Eccezione 6a.1a

#	Attore	Sistema
6a.1a.1	Modifica la ricorrenza di un evento pre-esistente	La ricorrenza è in corso non può essere modificata
Termine del caso d'uso		

Estensione 6b

#	Attore	Sistema
6b.1	Elimina una ricorrenza di un evento esistente	Eliminata la ricorrenza di un evento esistente

Eccezione 6b.1a

#	Attore	Sistema
6b.1a.1	Elimina una ricorrenza di un evento esistente	La ricorrenza è in corso e non può essere eliminata
Termine del caso d'uso		

Modello di dominio

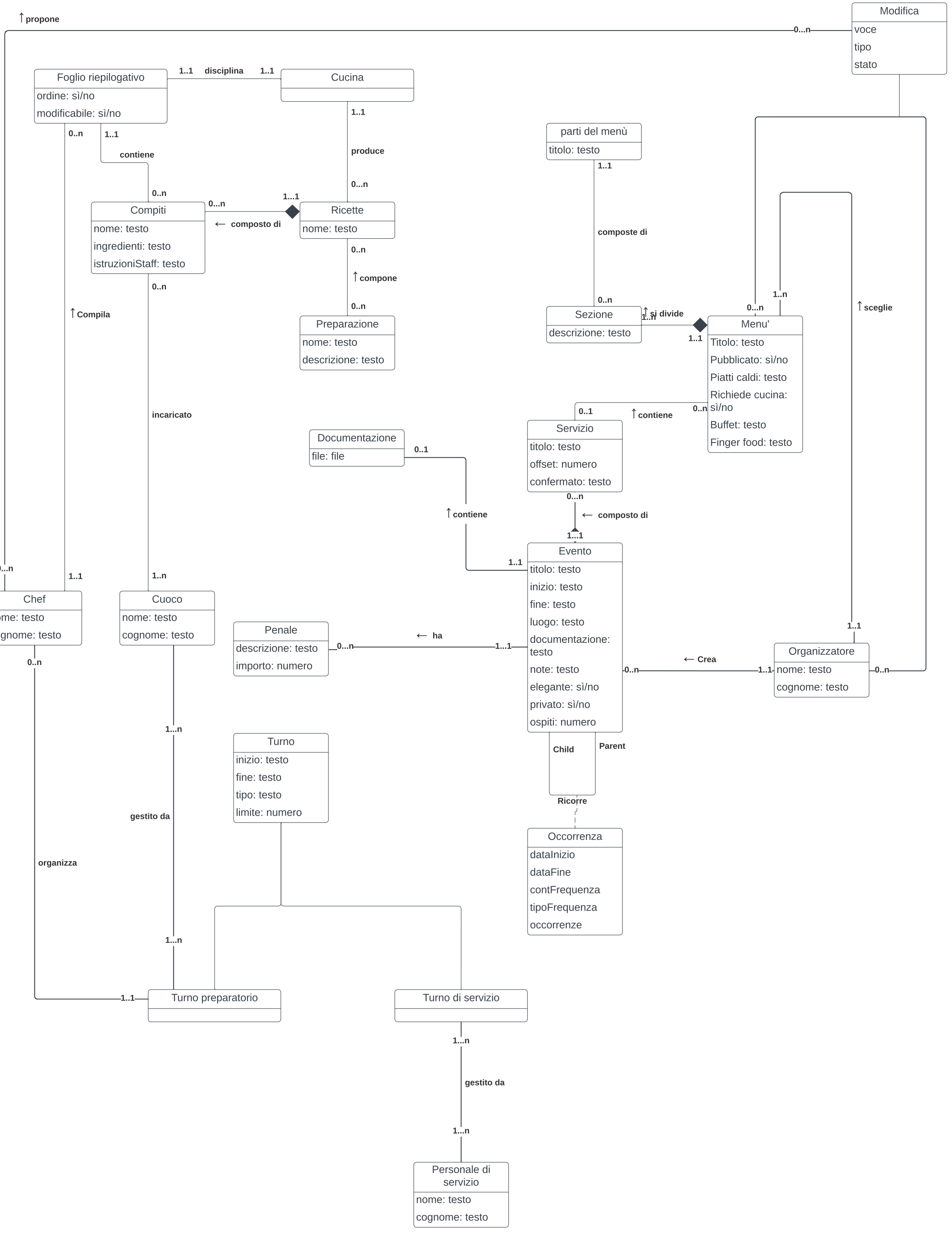
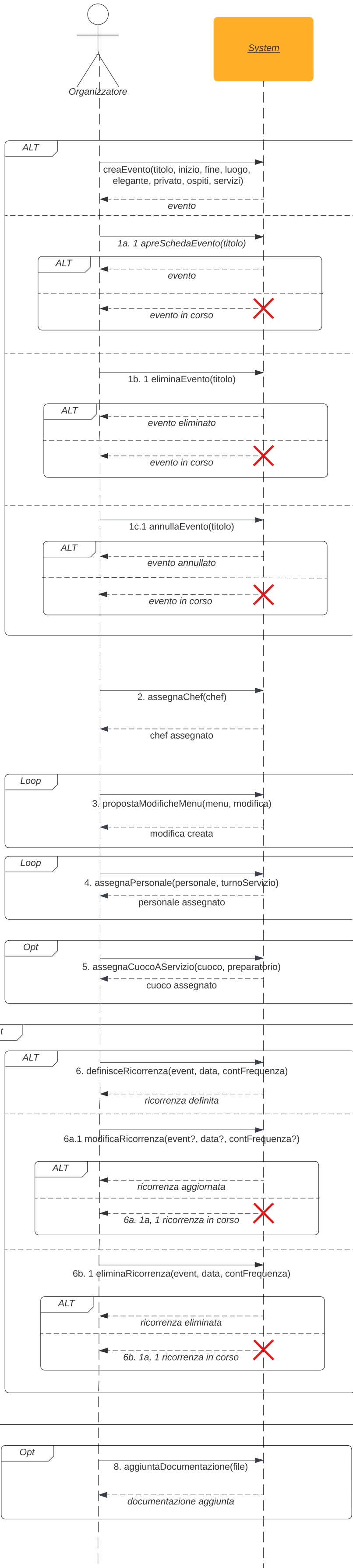

propone
Modifica
voce
tipo
stato
0...n
Foglio riepilogativo
ordine: sì/no
modificabile: sì/no
1..1
disciplina
1..1
Cucina
1..1
produce
0...n
0..n
1..1
contiene
0..n
Compiti
nome: testo
ingredienti: testo
istruzioniStaff: testo
0...n
composto di
1...1
Ricette
nome: testo
0..n
compone
0..n
Preparazione
nome: testo
descrizione: testo
0..n
Compila
incaricato
parti del menù
titolo: testo
1..1
composte di
0..n
Sezione
descrizione: testo
si divide
1..n
1..1
0...n
1..n
Menu'
Titolo: testo
Pubblicato: sì/no
Piatti caldi: testo
Richiede cucina: sì/no
Buffet: testo
Finger food: testo
sceglie
0..1
contiene
0..n
Servizio
titolo: testo
offset: numero
confermato: testo
Documentazione
file: file
0..1
contiene
0...n
composto di
1...1
Evento
titolo: testo
inizio: testo
fine: testo
luogo: testo
documentazione: testo
note: testo
elegante: sì/no
privato: sì/no
ospiti: numero
1..1
Child
Parent
Ricorre
Occorrenza
dataInizio
dataFine
contFrequenza
tipoFrequenza
occorrenze
0...n
1..1
Chef
nome: testo
cognome: testo
1..n
Cuoco
nome: testo
cognome: testo
Penale
descrizione: testo
importo: numero
0...n
ha
1...1
0..n
Crea
1..1
1..1
Organizzatore
nome: testo
cognome: testo
0..n
0..n
organizza
1...n
gestito da
Turno
inizio: testo
fine: testo
tipo: testo
limite: numero
1...n
gestito da
1..1
Turno preparatorio
Turno di servizio
1...n
gestito da
1...n
Personale di servizio
nome: testo
cognome: testo

Diagramma di sequenza di sistema

SSD Gestire gli eventi

Massimiliano Bettarello, Massimo Mendozza,, | February 3, 2024



Modello di dominio

Contratti

Pre-condizione generale: l'attore è identificato con un'istanza *or* di Organizzatore

1. creaEvento(titolo : testo, inizio: testo, fine: testo, luogo: testo, elegante: booleano, privato: booleano, ospiti: numero, servizi: Servizio..*)

pre-condizioni:

- *L'Organizzatore or è stato assegnato a un evento*

post-condizioni:

- è stata creata un'istanza *e* di Evento
- *e.titolo* = titolo
- *e.inizio* = inizio
- *e.fine* = fine
- *e.luogo* = luogo
- *e.elegante* = elegante
- *e.privato* = privato
- *e.ospiti* = ospiti
- *or* è proprietario di *e*
- Per ogni elemento serv di servizi viene creata un'istanza *s* di Servizio ed viene associata ad *e*

2. assegnaChef (chef : Chef)

pre-condizioni:

- è in corso la definizione di una evento *e*
- non esiste un *e'* evento in cui *e'.inizio* = *e.inizio* dove chef è assegnato a *e'*

post-condizioni:

- chef è assegnato ad *e*

3. propostaModificheMenu (menu: Menu, modifica: Modifica...*)

pre-condizioni:

- esiste un'istanza *m* di menu tale che menu = *m*;

post-condizioni:

- è creata un'istanza della modifica chiamata *mod*
- L'istanza *mod* è associata a *m*

4. assegnaPersonale (personale : Personale...*, turnoServizio: Turno)

pre-condizioni:

- è in corso la definizione di un evento *e*
- esiste un'istanza *sse* di Turno di servizio dove turnoServizio = *sse*

post-condizioni:

- personale è assegnato a *sse*

5. assegnaCuocoAServizio (cuoco : Personale, preparatorio: Turn)

pre-condizioni:

- è in corso la definizione di una Evento *e*
- esiste un'istanza *sp* di Turno preparatorio dove preparatorio= *sp*

post-condizioni:

- cuoco è assegnato a preparatorio

6. definisceRicorrenza (event: Evento, data: testo, contFrequenza: numero)

pre-condizioni:

- L'Organizzatore *or* è stato assegnato a un evento

post-condizioni:

- è stata creata un'istanza *r* di Ricorrenza
- *r.event* = event
- *r.data* = data
- *r.contFrequenza* = contFrequenza

7. aggiuntaDocumentazione (file: file)

pre-condizioni:

- è in corso la definizione di una evento *e*
- se l'orario attuale - *e.inizio* è maggiore di *s.fineOffset*

post-condizioni:

- è stata creata un'istanza *d* di Documentazione
- *d.file* = file

Contratti delle estensioni

1a. apreSchedaEvento (titolo: testo)

pre-condizioni:

- *or* deve essere proprietario di evento
- la scheda evento non è in uso

1b. eliminaEvento (titolo: testo)

pre-condizioni:

- *or* deve essere proprietario di evento
- la scheda evento non è in uso

post-condizioni:

Se *or* è proprietario di evento e evento non è in uso:

- evento è eliminata.

1c annullaEvento (evento : evento)

pre-condizioni:

- *or* deve essere proprietario di evento
- la scheda evento non è in uso

post-condizioni:

- evento è annullato.

6a. modificaRicorrenza (event?: Evento, data? : data, contFrequenza? : numero)

pre-condizioni:

- è in corso la definizione di una ricorrenza *r*

post-condizioni:

- *[se* event *è specificato]*: *r.event* = event
- *[se* data *è specificato]*: *r.data* = data
- *[se* contFrequenza *è specificato]*: *r.contFrequenza* = contFrequenza

6b. eliminaRicorrenza (event: Evento, data: testo, contFrequenza: numero)

pre-condizioni:

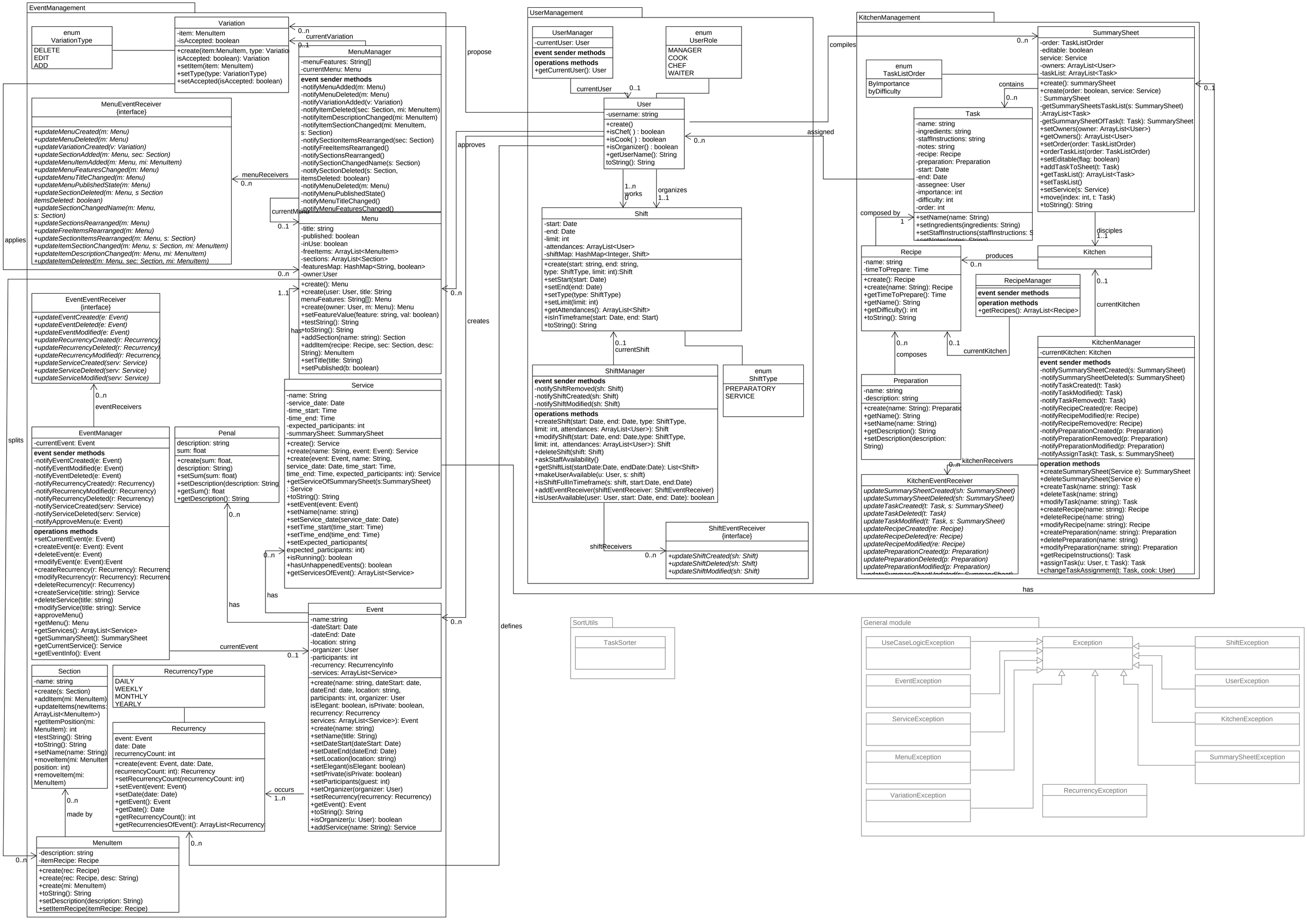
- è in corso la definizione di una ricorrenza *r*

post-condizioni:

- *r.evento*, *r.data*, *r.contFrequenza* sono eliminati dall'oggetto *r*

Design Class Diagram

Design Class Diagram (DCD)



System Sequence Diagram

1

CatERingAppManager.eventManager: EventManager

CatERingAppManager.userManager: UserManager

receivers: EventEventReceiver

createEvent(e)

getCurrentUser()

user

alt
[!user.isManager()]

throw UseCaseLogicException

create(name, dateStart, dateEnd, location, participants, organizer, recurrency, services, documentation, isElegant, isPrivate)

e: :Event

this.name = name

this.dateStart = dateStart

this.dateEnd = dateEnd

this.location = location

this.participants = participants

this.organizer = organizer

this.recurrency = recurrency

this.documentation = documentation

this.isElegant = isElegant

this.isPrivate = isPrivate

create

Services: ArrayList<Service>

for each
service in services

create(service)

service

servizice: :Service

this.event = service.event

this.name = service.name

this.proposed_menu_id = service.proposed_menu_id

this.approved_menu_id = service.approved_menu_id

this.service_date = service.service_date

this.time_start = service.time_start

this.time_end = service.time_end

this.expected_participants = service.expected_participants

add(service)

this.services = Services

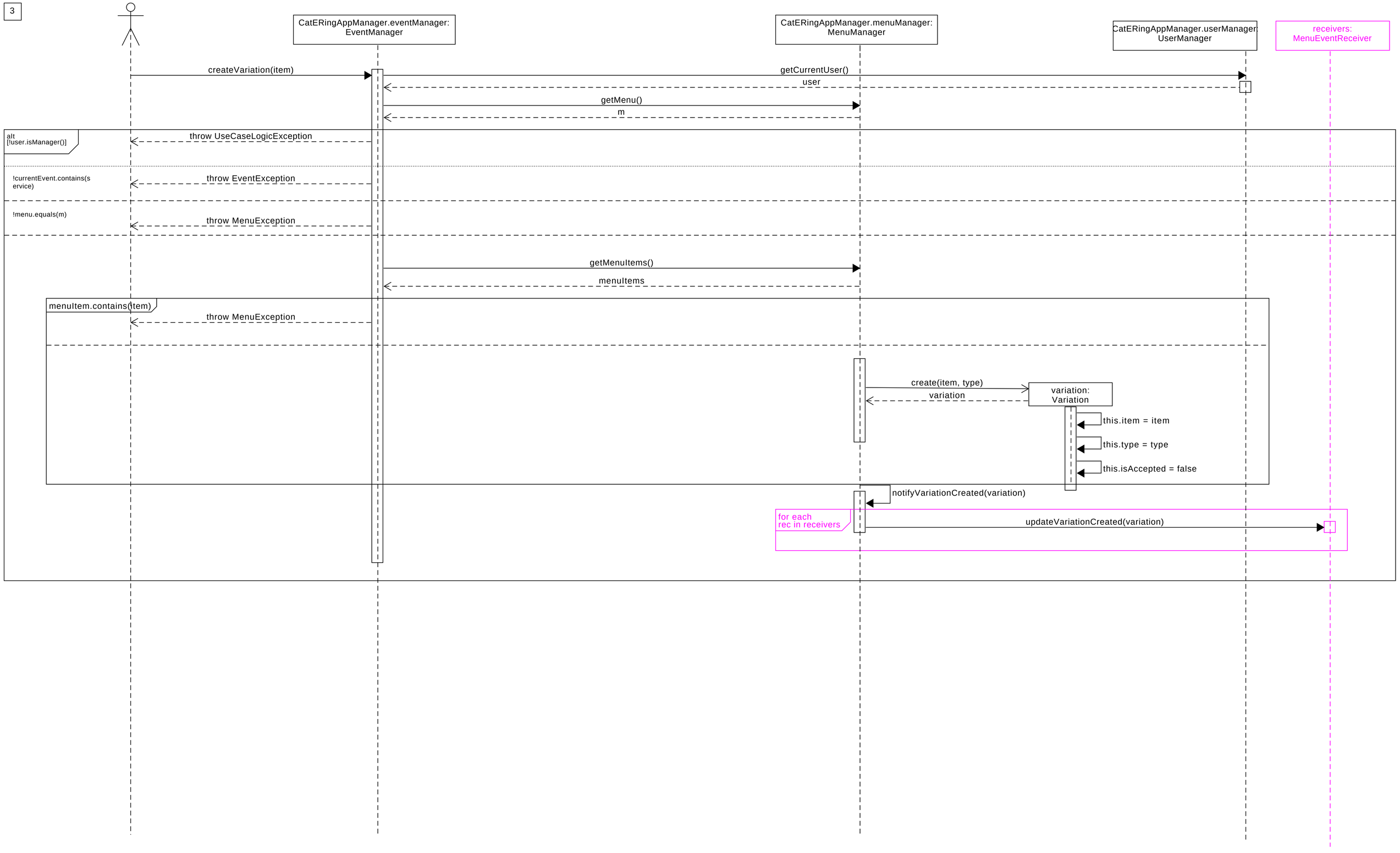
setCurrentEvent(e)

notifyEventCreated(e)

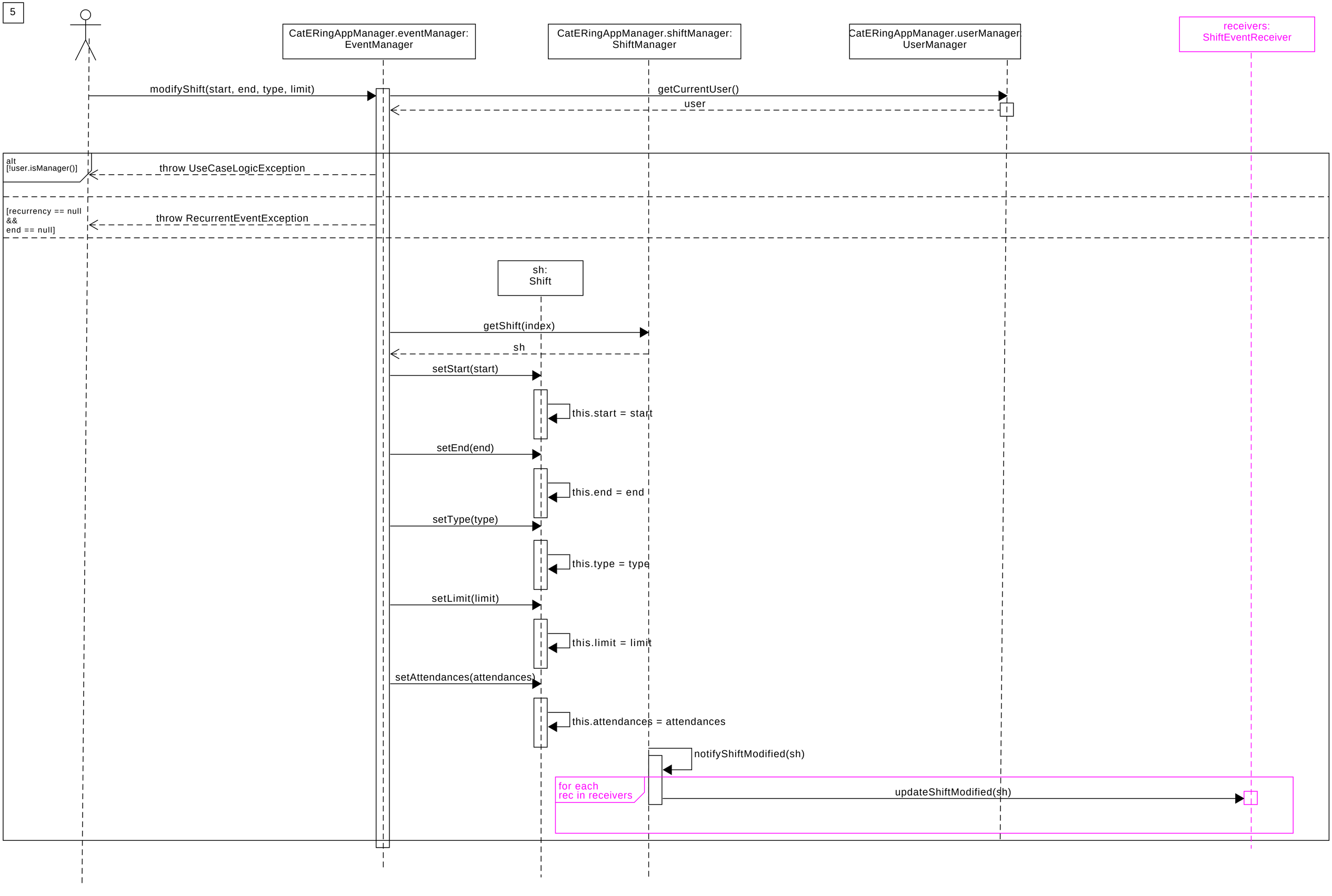
for each
rec in receivers

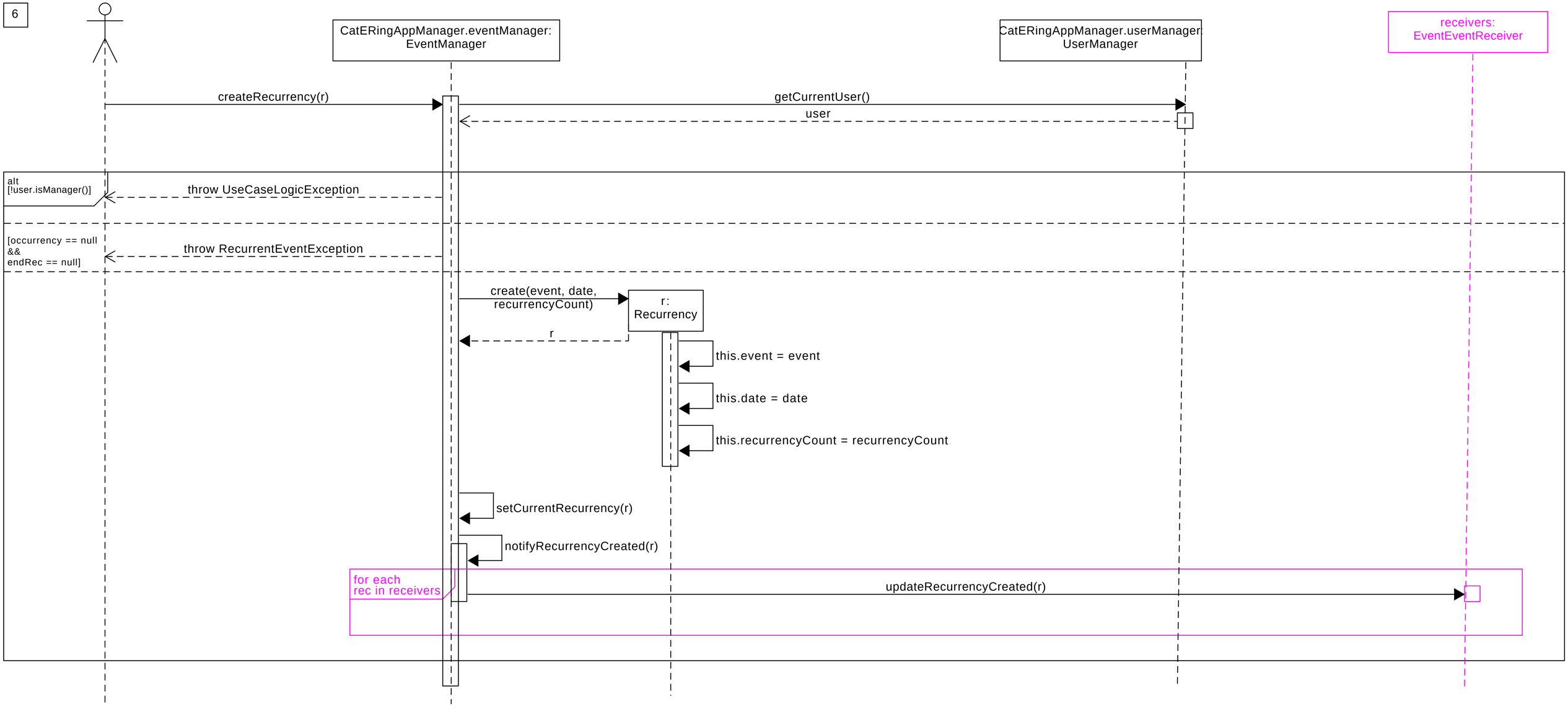
updateEventCreated(e)

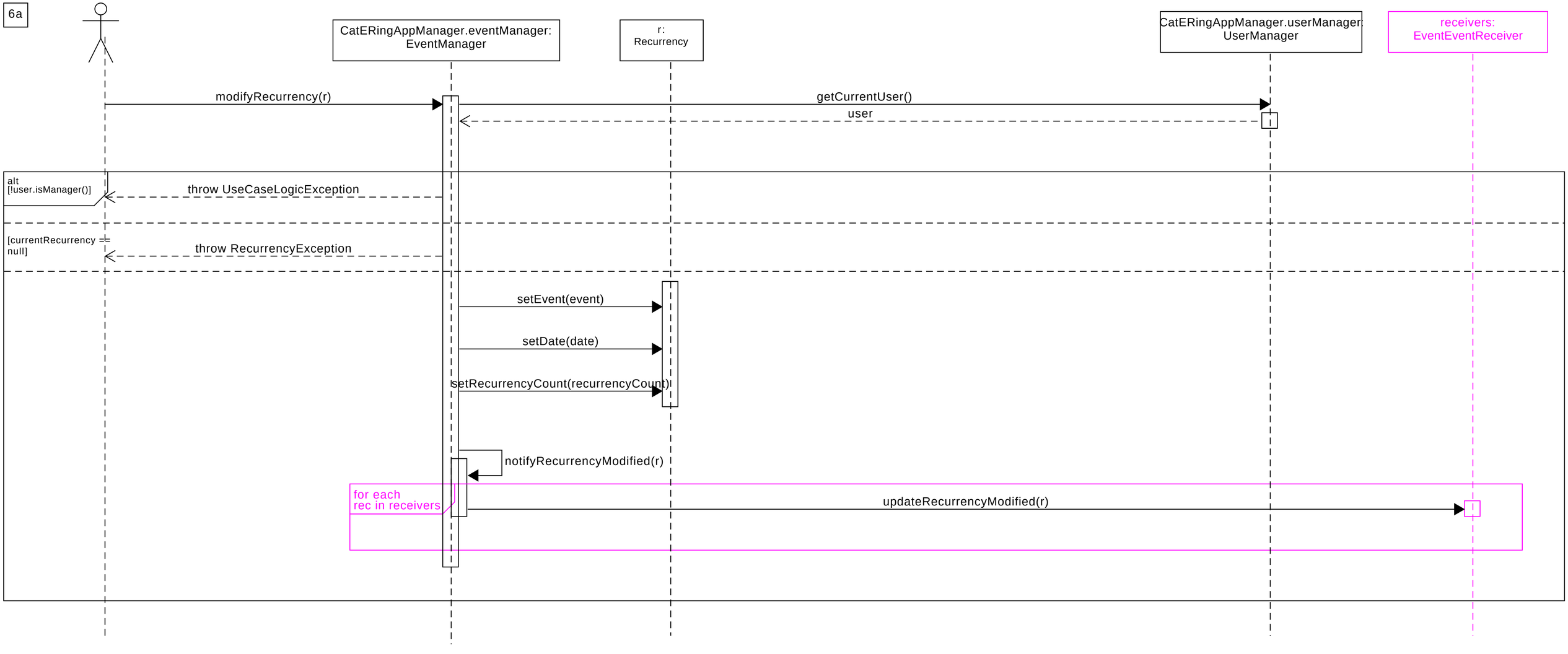
3


CatERingAppManager.eventManager: EventManager
CatERingAppManager.menuManager: MenuManager
CatERingAppManager.userManager: UserManager
receivers: MenuEventReceiver
createVariation(item)
getCurrentUser()
user
getMenu()
m
alt
[!user.isManager()]
throw UseCaseLogicException
!currentEvent.contains(service)
throw EventException
!menu.equals(m)
throw MenuException
getMenuItems()
menuItems
menuItem.contains(item)
throw MenuException
create(item, type)
variation
variation: Variation
this.item = item
this.type = type
this.isAccepted = false
notifyVariationCreated(variation)
for each
rec in receivers
updateVariationCreated(variation)

5


CatERingAppManager.eventManager: EventManager
CatERingAppManager.shiftManager: ShiftManager
CatERingAppManager.userManager: UserManager
receivers: ShiftEventReceiver
modifyShift(start, end, type, limit)
getCurrentUser()
user
alt
[!user.isManager()]
throw UseCaseLogicException
[recurrency == null
&&
end == null]
throw RecurrentEventException
sh: Shift
getShift(index)
sh
setStart(start)
this.start = start
setEnd(end)
this.end = end
setType(type)
this.type = type
setLimit(limit)
this.limit = limit
setAttendances(attendances)
this.attendances = attendances
notifyShiftModified(sh)
for each
rec in receivers
updateShiftModified(sh)


6
CatERingAppManager.eventManager:
EventManager
CatERingAppManager.userManager:
UserManager
receivers:
EventEventReceiver
createRecurrency(r)
getCurrentUser()
user
alt
[!user.isManager()]
throw UseCaseLogicException
[occurrency == null
&&
endRec == null]
throw RecurrentEventException
create(event, date,
recurrencyCount)
r:
Recurrency
r
this.event = event
this.date = date
this.recurrencyCount = recurrencyCount
setCurrentRecurrency(r)
notifyRecurrencyCreated(r)
for each
rec in receivers
updateRecurrencyCreated(r)


6a
CatERingAppManager.eventManager:
EventManager
r:
Recurrency
CatERingAppManager.userManager:
UserManager
receivers:
EventEventReceiver
modifyRecurrency(r)
getCurrentUser()
user
alt
[!user.isManager()]
throw UseCaseLogicException
[currentRecurrency ==
null]
throw RecurrencyException
setEvent(event)
setDate(date)
setRecurrencyCount(recurrencyCount)
notifyRecurrencyModified(r)
for each
rec in receivers
updateRecurrencyModified(r)

7

CatERingAppManager.eventManager: EventManager

CatERingAppManager.userManager: UserManager

receivers: EventEventReceiver

aggiuntaDocumentazione (file)

getCurrentUser()

user

alt
[!user.isManager()]

throw UseCaseLogicException

se: :Service

currentDate: :LocalDateTime

getCurrentService()

se

getCurrentTime()

currentDateTime

e: :Event

getEvent()

e

getDocumentation()

d

d: Documentatio

[currentDateTime - s.timeEnd > serviceEndOffset]

this.file = file

setCurrentEvent(e)

notifyEventModifie(e)

for each
rec in receivers

updateEventModified(e)