Casi d'Uso Gestire gli Eventi

Informazioni generali

Nome caso d'uso: Gestire gli eventi
Portata: Sistema
Livello: Obiettivo utente
Attore primario: Organizzatore
Parti Interessate: Chef, Cuoco, Personale di Servizio, Cliente
Pre-condizioni: L'attore deve essere autenticato come organizzatore.
Garanzie di successo o post-condizioni: L'evento viene organizzato assieme ai suoi servizi, vengono scelti lo chef, i cuochi, il personale e i menù per l'evento.

Scenario principale di successo

#	Attore	Sistema
1	Consulta lo stato della cucina (per definire se è possibile organizzare l'evento)	Fornisce stato della cucina
	Se l'evento non è organizzabile termina il caso d'uso. Altrimenti passo 2	
2	Genera la scheda evento, specificando il cliente, la data di inizio, il numero previsto di partecipanti e la location[1]	Registra una nuova scheda evento con i dati forniti. L'evento è in stato "pianificato".
3	Opzionalmente inserisce un servizio specificando la sua tipologia, il giorno di inizio[2], l'ora di inizio, l'ora di fine, opzionalmente il luogo[3]. Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze	Registra il servizio inserito con i relativi parametri forniti. Eventualmente applica le modifiche anche per le successive occorrenze dell'evento.
	Ripete dal passo 3 finché non è soddisfatto.	
4	Opzionalmente inserisce delle note particolari all'evento	Registra le note particolari relative all'evento
5	Opzionalmente associa uno chef all'evento. Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze	Registra lo chef associato. Eventualmente applica le modifiche anche per le successive occorrenze dell'evento.

[1] L'organizzatore sceglie la location tra le disponibili nel caso non venga fornita dal cliente
[2] Il giorno di inizio espresso come numero giorni dall'inizio dell'evento
[3] L'area della location in cui si svolge il servizio

6	Opzionalmente conferma un menù dello chef per un servizio	Registra la conferma, il servizio va in stato "confermato"[4] e se tutti i menu sono stati confermati l'evento va in stato "in corso"[5]
	Ripete il passo 6 se ci sono altri menù da approvare.	
7	Opzionalmente crea un compito per un servizio prenotando un membro del personale per un turno per cui ha dato disponibilità e opzionalmente assegnandogli un ruolo	Registra prenotazione membro del personale e il relativo ruolo se specificato
	Ripete passo 7 finché non è soddisfatto	
8	Opzionalmente richiede l'assistenza di un cuoco per un servizio	Registra il cuoco inserito per il servizio scelto
	Ripetere dal passo 8 per ogni servizio	
9	Opzionalmente rende l'evento ricorrente specificando la frequenza con cui l'evento si ripete e il numero di volte in cui si ripete	Rendi l'evento ricorrente generando tutte le occorrenze richieste
	Termina il caso d'uso	

Eccezione 2a

#	Attore	Sistema
2a.1	Genera la scheda evento, specificando il cliente, la data di inizio, il numero previsto di partecipanti e la location[6]	La data di inizio è già passata
	Termina caso d'uso	

[4]Stato servizio:
- "Da confermare": il menu del servizio non è stato ancora confermato
- "Confermato": se il menu del servizio è stato confermato
- "Annullato": il servizio è stato annullato
- "Terminato": il servizio è terminato

[5]Stato evento:
- "Pianificato": un evento futuro
- "In corso": se tutti i menu dei servizi degli eventi sono stati confermati
- "Annullato": ogni suo servizio viene annullato
- "Terminato": l'evento è terminato

[6] L'organizzatore sceglie la location tra le disponibili nel caso non venga fornita dal cliente

Eccezione 5a

#	Attore	Sistema
5a.1	Opzionalmente associa uno chef all'evento	Lo chef scelto per la data dell'evento è già occupato in un altro evento
	Termina caso d'uso	

Eccezione 6a

#	Attore	Sistema
6a.1	Opzionalmente conferma un menù dello chef per un servizio	Il menù è già stato confermato o il servizio non ha un menù associato
	Termina caso d'uso	

Eccezione 8a

#	Attore	Sistema
8a.1	Opzionalmente richiede l'assistenza di un cuoco per un servizio	Il cuoco non ha dato la disponibilità per il turno scelto
	Termina caso d'uso	

Eccezione 9a

#	Attore	Sistema
9a.1	Opzionalmente rende l'evento ricorrente specificando la frequenza con cui l'evento si ripete e il numero di volte in cui si ripete	L'evento è già ricorrente
	Termina caso d'uso	

Estensione 1a

#	Attore	Sistema
1a.1	Apre un evento esistente “pianificato” o “in corso” per annullarlo. Se l’evento è ricorrente opzionalmente annulla anche le successive occorrenze.	Annulla l’evento selezionato ed eventualmente anche le successive occorrenze dell’evento
	Termina il caso d’uso	

Estensione 1b

#	Attore	Sistema
1b.1	Apre un evento esistente per eliminarlo. Se l’evento è ricorrente opzionalmente elimina anche le successive occorrenze	Elimina evento selezionato ed eventualmente anche le successive occorrenze dell’evento

Estensione 1c

#	Attore	Sistema
1c.1	Apre un evento esistente per terminarlo	Termina evento esistente
1c.2	Opzionalmente aggiunge delle note finali di un evento e opzionalmente relativi documenti	Registra le note finali dell’evento
	Termina caso d’uso	

Estensione 1d

#	Attore	Sistema
1d.1	Aggiunge una rimanenza prodotta da un’attività di cucina di un certo servizio di un evento	Salva rimanenza di un attività di cucina
	Ripete passo 1d.1 per tutte le rimanenze	
	Termina caso d’uso	

Eccezione 1d.1a

#	Attore	Sistema
1d.1 a.1	Aggiunge una rimanenza prodotta da un'attività di cucina di un certo servizio di un evento	Il servizio non è ancora terminato
	Termina caso d'uso	

Estensione (1-2)a

#	Attore	Sistema
(1-2)a.1	Apre evento esistente "pianificato" o "in corso" per modificarlo	Fornisci la scheda evento
(1-2)a.2	Opzionalmente modifica la data di inizio e/o il numero previsto di partecipanti e/o la location.	Registra le modifiche scelte per l'evento
	Prosegue al passo 3 dello scenario principale oppure termina il caso d'uso	

Eccezione (1-2)a.2a

#	Attore	Sistema
(1-2)a. 2a.1	Opzionalmente modifica la data di inizio e/o il numero previsto di partecipanti e/o la location.	La data di inizio è già passata
	Termina caso d'uso	

Estensione 3a

#	Attore	Sistema
3a.1	Sceglie un servizio dell'evento per modificare la sua tipologia e/o il giorno di inizio e/o l'ora inizio e/o l'ora di fine e/o il luogo.	Registra le modifiche effettuate al servizio dell'evento ed eventualmente anche per le successive ricorrenze dell'evento.

	Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze	
	Prosegue al passo 4 dello scenario principale oppure termina il caso d'uso	

Eccezione 3a.1a

#	Attore	Sistema
3a.1a.1	Sceglie un servizio dell'evento per modificare la sua tipologia e/o il giorno di inizio e/o l'ora inizio e/o l'ora di fine e/o il luogo. Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze	Il servizio è già stato annullato o è terminato o la data di inizio è già passata
	Termina caso d'uso	

Estensione 3a.1a

#	Attore	Sistema
3a.1a.1	Sceglie un servizio dell'evento per annullarlo. Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze.	Annulla servizio dall'evento liberando il personale prenotato. Avvisa di una possibile penale. Eventualmente applica le modifiche anche per le successive occorrenze dell'evento.

Eccezione 3a.1a.1a

#	Attore	Sistema
3a.1a.1a.1	Sceglie un servizio dell'evento per annullarlo. Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze.	Il servizio è già stato annullato o è terminato o la data di inizio è già passata
	Termina caso d'uso	

Estensione 3a.1b

#	Attore	Sistema
3a.1 b.1	Sceglie un servizio dell'evento per eliminarlo. Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze.	Rimuove il servizio indicato. Eventualmente applica le modifiche anche per le successive occorrenze dell'evento.

Eccezione 3a.1b.1a

#	Attore	Sistema
3a. 1b.1 a.1	Sceglie un servizio dell'evento per eliminarlo. Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze.	Il servizio è già stato annullato o è terminato o la data di inizio è già passata
	Termina caso d'uso	

Estensione 5a

#	Attore	Sistema
5a.1	Modifica lo chef assegnato all'evento. Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze.	Registra la modifica effettuata. Eventualmente applica le modifiche anche per le successive occorrenze dell'evento.
	Prosegui al passo 6 dello scenario principale oppure termina il caso d'uso	

Eccezione 5a.1a

#	Attore	Sistema
5a. 1a.1	Modifica lo chef assegnato all'evento. Se l'evento è ricorrente opzionalmente applica le modifiche anche alle successive occorrenze.	L'evento non ha ancora uno chef assegnato
	Termina caso d'uso	

Estensione 6a

#	Attore	Sistema
6a.1	Propone modifiche al menù scelto dallo chef[7]	Registra le modifiche del menu

Eccezione 6a.1a

#	Attore	Sistema
6a. 1a.1	Propone modifiche al menù scelto dallo chef	Il menù è già stato confermato o il servizio non ha un menù associato
	Termina caso d'uso	

Estensione 7a

#	Attore	Sistema
7a.1	Libera un membro del personale prenotato per un compito di un servizio.	Registra la scelta effettuata.
	Ripeti passo 7a.1 finché non soddisfatto	
	Prosegui al passo 8 dello scenario principale oppure termina il caso d'uso	

Estensione 7b

#	Attore	Sistema
7b.1	Assegna un membro del personale a un compito esistente di un servizio.	Registra la scelta effettuata.
	Prosegui al passo 8 dello scenario principale oppure termina il caso d'uso	

Eccezione 7b.1a

#	Attore	Sistema
7b. 1a.1	Assegna un membro del personale a un compito esistente di un servizio.	Il membro del personale non ha dato la disponibilità per il turno del compito di servizio
	Termina caso d'uso	

[7] Lo chef accettando le modifiche può decidere se applicarle a tutti i menu degli eventi successivi

Eccezione 7b.1b

#	Attore	Sistema
7b.1b.1	Assegna un membro del personale a un compito esistente di un servizio.	Il membro del personale è già stato prenotato per il compito di servizio
	Termina caso d'uso	

Estensione 7c

#	Attore	Sistema
7c.1	Modifica il ruolo di un compito di un servizio	Registra la scelta effettuata.
	Prosegui al passo 8 dello scenario principale oppure termina il caso d'uso	

Estensione 8a

#	Attore	Sistema
8a.1	Libera un cuoco prenotato per un servizio.	Registra la scelta effettuata.
	Prosegui al passo 9 dello scenario principale oppure termina il caso d'uso	

Estensione 9a

#	Attore	Sistema
9a.1	Opzionalmente rende l'evento ricorrente specificando la frequenza e la data di terminazione	Rendi l'evento ricorrente generando tutte le occorrenze richieste

Eccezione 9a.1a

#	Attore	Sistema
9a.1a.1	Opzionalmente rende l'evento ricorrente specificando la frequenza e la data di terminazione	La data di terminazione inserita è già passata
	Termina caso d'uso	

Estensione 9b

#	Attore	Sistema
9b.1	Modifica la periodicità di un evento specificando l'occorrenza da usare come base e impostando la nuova frequenza e il numero di ripetizioni	Sostituisci le successive occorrenze di evento con le nuove definite dalla nuova periodicità
	Termina il caso d'uso	

Estensione 9c

#	Attore	Sistema
9c.1	Modifica la periodicità di un evento specificando l'occorrenza da usare come base e impostando la nuova frequenza e la data di terminazione	Sostituisci le successive occorrenze di evento con le nuove definite dalla nuova periodicità
	Termina il caso d'uso	

Eccezione 9c.1a

#	Attore	Sistema
9c.1a.11	Modifica la periodicità di un evento, specificando l'occorrenza da usare come base e impostando la nuova frequenza e la data di terminazione	La data di terminazione inserita è già passata
	Termina caso d'uso	

Eccezione (9b.1-9c.1)a

#	Attore	Sistema
9b.1a.1a.1	Modifica la periodicità di un evento.	L'evento non è ricorrente
	Termina caso d'uso	

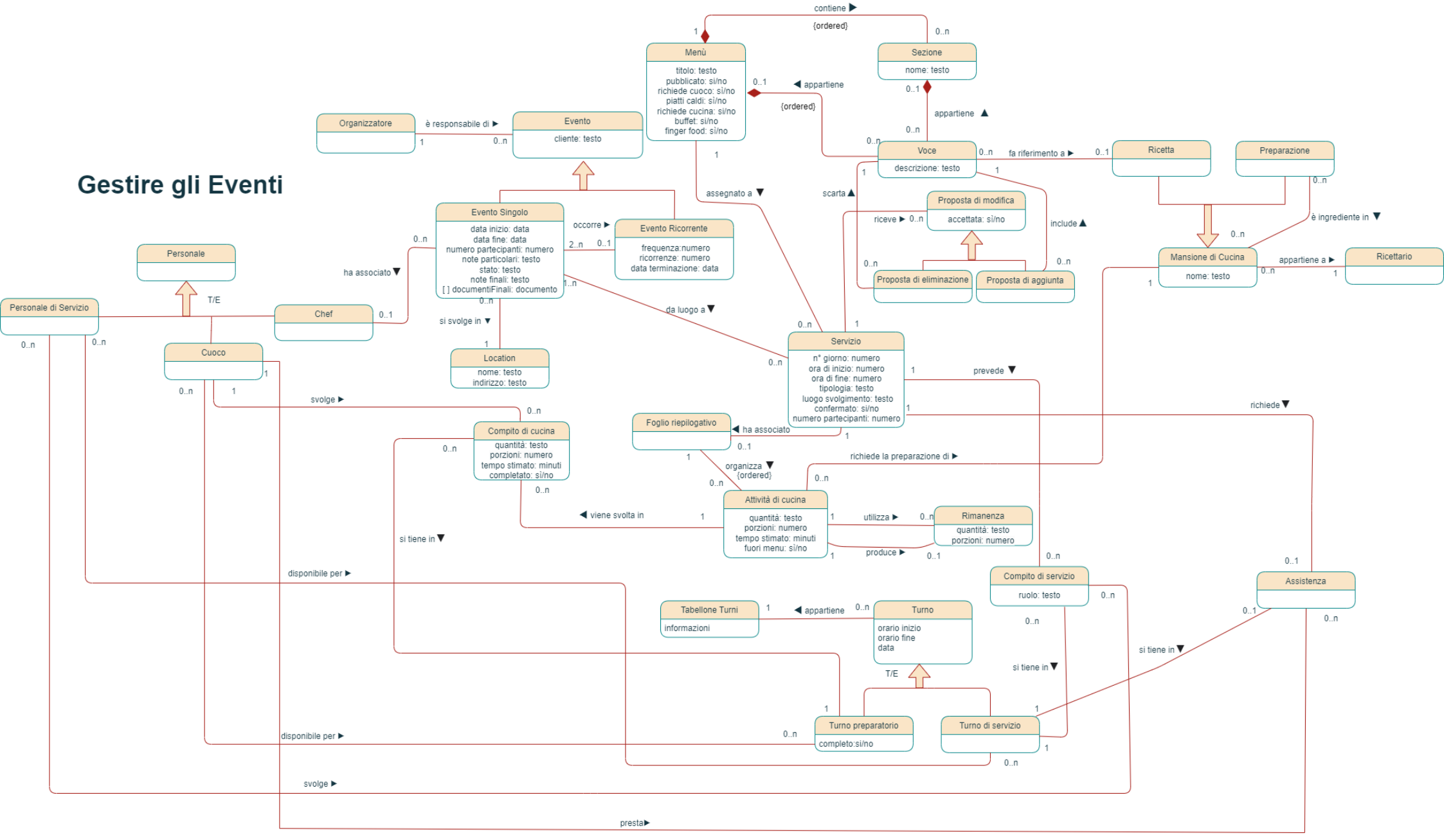
Eccezione 9d.1a

#	Attore	Sistema
9d.1a	Modifica la periodicità di un evento, specificando l'occorrenza da usare come base e impostando la nuova frequenza e la data di terminazione	L'occorrenza da usare come base non occorre nello stesso evento ricorrente dell'evento corrente
	Termina caso d'uso	

Eccezione (1a.1 - 1b.1 - 1c.1 - 1d.1 - (1-2)a.1)a

#	Attore	Sistema
(1a.1-1b.1-1c.1-1d.1-(1-2)a.1)a.1	Apre un evento	L'utente non ha i permessi adatti per effettuare la modifica
	Termina il caso d'uso	

Gestire gli Eventi



Organizzatore

SSD per l'UC Gestire gli Eventi
Scenario principale

:Sistema

1. statoCucina()

ALT

statoCucina

statoCucina

2. creaSchedaEvento(cliente, dataInizio, dataFine, nPartecipanti, location)

ok

LOOP

3. aggiungiServizio(tipologia, giornoInizio, oraInizio, oraFine, luogoSvolgimento?, applicaSuccessivi)

ok

OPT

4. aggiungiNoteParticolari(note)

ok

OPT

5. associaChef(chef, applicaSuccessivi)

ok

LOOP

6. confermaMenu(servizio)

ok

LOOP

7. generaCompitoDiServizio(servizio, membroPersonale, turnoDiServizio, ruolo?)

ok

LOOP

8. richiediAssistenzaCuoco(servizio, cuoco, turnoDiServizio)

ok

OPT

9.setEventoRicorrente(frequenza, ripetizioni)

ok

SSD per l'UC Gestire gli Eventi
Estensioni ed eccezioni del passo 2, 5, 6 e 8

Organizzatore

:Sistema

1. statoCucina()

ALT
statoCucina
statoCucina

2. creaSchedaEvento(cliente, dataInizio, dataFine, nPartecipanti, location)

ALT
ok
2a.1 La data di inizio è già passata

LOOP
3. aggiungiServizio(tipologia, giornoInizio, oraInizio, oraFine, luogoSvolgimento?, applicaSuccessivi)
ok

OPT
4. aggiungiNoteParticolari(note)
ok

ALT
OPT
5. associaChef(chef, applicaSuccessivi)
ALT
ok
5a.1 Lo chef scelto per la data dell'evento è già occupato in un altro evento
5a. modifcaChefAssegnato(chef, applicaSuccessivi)
ALT
ok
ok
5a.1a.1 L'evento non ha ancora uno chef assegnato

ALT
LOOP
6. confermaMenu(servizio)
ALT
ok
6a.1 Il menù è già stato confermato o il servizio non ha un menù associato
6a. proponiModifica(servizio, voce, operazione)
ALT
ok
6a.1a Il menù è già stato confermato o il servizio non ha un menù associato

LOOP
7. generaCompitoDiServizio(servizio, membroPersonale, turnoDiServizio, ruolo?)
ok

ALT
LOOP
8. richiediAssistenzaCuoco(servizio, cuoco, turnoDiServizio)
ALT
ok
8a.1 il cuoco non ha dato la disponibilità per il turno scelto
8a. liberaCuoco(servizio, cuoco)
ALT
ok
ok

OPT
9.setEventoRicorrente(frequenza, ripetizioni)
ok

Organizzatore

SSD per l'UC Gestire gli Eventi
Estensioni ed eccezioni del passo 1

:Sistema

ALT

ALT

1. statoCucina()

ALT

statoCucina

statoCucina

1a.1. annullaEvento(eventoSingolo, applicaSuccessivi)

ALT

ok

(1a.1 - 1b.1 - 1c.1 - 1d.1 - (1-2)a.1)a L'utente non ha i permessi adatti per effettuare la modifica

1b.1. eliminaEvento(eventoSingolo, applicaSuccessivi)

ALT

ok

(1a.1 - 1b.1 - 1c.1 - 1d.1 - (1-2)a.1)a L'utente non ha i permessi adatti per effettuare la modifica

1c.1. terminaEvento(eventoSingolo)

ALT

ok

(1a.1 - 1b.1 - 1c.1 - 1d.1 - (1-2)a.1)a L'utente non ha i permessi adatti per effettuare la modifica

1c.2. aggiungiNoteFinali(noteFinali, documentiFInali)

ok

LOOP

1d.1. aggiungiRimanenza(attivitàDiCucina, rimanenza)

ALT

ok

ok

1d.1a.1 il servizio non è terminato

(1a.1 - 1b.1 - 1c.1 - 1d.1 - (1-2)a.1)a L'utente non ha i permessi adatti per effettuare la modifica

2. creaSchedaEvento(cliente, dataInizio, dataFine, nPartecipanti, location)

ok

(1-2)a.1 apriEvento(evento)

ok

(1-2)a.2. modificaEvento(dataInizio?, nPartecipanti?, location?, chef?)

ALT

ok

(1-2)a-2a.1 La data di inizio è già passata

(1a.1 - 1b.1 - 1c.1 - 1d.1 - (1-2)a.1)a L'utente non ha i permessi adatti per effettuare la modifica

LOOP

3. aggiungiServizio(tipologia, giornoInizio, oraInizio, oraFine, luogoSvolgimento?, applicaSuccessivi)

ok

OPT

4. aggiungiNoteParticolari(note)

ok

OPT

5. associaChef(chef, applicaSuccessivi)

ok

LOOP

6. confermaMenu(servizio)

ok

LOOP

7. generaCompitoDiServizio(servizio, membroPersonale, turnoDiServizio, ruolo?)

ok

LOOP

8. richiediAssistenzaCuoco(servizio, cuoco, turnoDiServizio)

ok

OPT

9.setEventoRicorrente(frequenza, ripetizioni)

ok

Organizzatore

SSD per l'UC Gestire gli Eventi
Estensioni ed eccezioni del passo 3

:Sistema

1. statoCucina()

ALT

statoCucina

statoCucina

2. creaSchedaEvento(cliente, dataInizio, dataFine, nPartecipanti, location)

ok

ALT

LOOP

3. aggiungiServizio(tipologia, giornoInizio, oraInizio, oraFine, luogoSvolgimento?, applicaSuccessivi)

ok

ALT

3a.1. modificaServizio(servizio, tipologia?, giornoInizio, oraInizio?, oraFine?, luogoSvolgimento, applicaSuccessivi)

ALT

ok

3a.1a.1 Il servizio è già stato annullato o è terminato o la data di inizio è già passata

3a.1a.1. annullaServizio(servizio, applicaSuccessivi)

ALT

ok

3a.1a.1a.1 Il servizio è già stato annullato o è terminato o la data di inizio è già passata

3b.1a.1 eliminaServizio(servizio, applicaSuccessivi)

ALT

ok

23a.1b.1a.1 Il servizio è già stato annullato o è terminato o la data di inizio è già passata

OPT

4. aggiungiNoteParticolari(note)

ok

OPT

5. associaChef(chef, applicaSuccessivi)

ok

LOOP

6. confermaMenu(servizio)

ok

LOOP

7. generaCompitoDiServizio(servizio, membroPersonale, turnoDiServizio, ruolo?)

ok

LOOP

8. richiediAssistenzaCuoco(servizio, cuoco, turnoDiServizio)

ok

OPT

9.setEventoRicorrente(frequenza, ripetizioni)

ok

Organizzatore

SSD per l'UC Gestire gli Eventi
Estensioni ed eccezioni del passo 7

:Sistema

1. statoCucina()

ALT

statoCucina

statoCucina

2. creaSchedaEvento(cliente, dataInizio, dataFine, nPartecipanti, location)

ok

LOOP

3. aggiungiServizio(tipologia, giornoInizio, oraInizio, oraFine, luogoSvolgimento?, applicaSuccessivi)

ok

OPT

4. aggiungiNoteParticolari(note)

ok

OPT

5. associaChef(chef, applicaSuccessivi)

ok

LOOP

6. confermaMenu(servizio)

ok

ALT

LOOP

7. generaCompitoDiServizio(servizio, membroPersonale, turnoDiServizio, ruolo?)

ok

LOOP

7a.1 liberaMembroPersonale(compitoDiServizio, membroPersonale)

ALT

ok

ok

LOOP

7b.1 ssegnaMembroPersonale(membroPersonale, compitoDiServizio)

ALT

ok

ok

7d.1a.1 Il membro del personale non ha dato la disponibilità per il turno del compito di servizio

7b.1b.1 Il membro del personale è già stato prenotato per il compito di servizio

LOOP

7b.1 ssegnaMembroPersonale(membroPersonale, compitoDiServizio)

ALT

ok

ok

LOOP

8. richiediAssistenzaCuoco(servizio, cuoco, turnoDiServizio)

ok

OPT

9.setEventoRicorrente(frequenza, ripetizioni)

ok

SSD per l'UC Gestire gli Eventi
Estensioni ed eccezioni del passo 9

Organizzatore

:Sistema

1. statoCucina()

ALT

statoCucina

statoCucina

2. creaSchedaEvento(cliente, dataInizio, dataFine, nPartecipanti, location)

ok

LOOP

3. aggiungiServizio(tipologia, giornoInizio, oraInizio, oraFine, luogoSvolgimento?, applicaSuccessivi)

ok

OPT

4. aggiungiNoteParticolari(note)

ok

OPT

5. associaChef(chef, applicaSuccessivi)

ok

LOOP

6. confermaMenu(servizio)

ok

LOOP

7. generaCompitoDiServizio(servizio, membroPersonale, turnoDiServizio, ruolo?)

ok

LOOP

8. richiediAssistenzaCuoco(servizio, cuoco, turnoDiServizio)

ok

ALT

OPT

9.setEventoRicorrente(frequenza, ripetizioni)

ALT

ok

9a.1 L'evento è già ricorrente

OPT

9a.setEventoRicorrenteData(frequenza, dataTerminazione)

ALT

ok

9a.1 L'evento è già ricorrente

9b.1. modifcaRicorrenzaRipetizioni(frequenza, ripetizione)

ALT

ok

9b.1a.1 L'evento non è ricorrente

9b.1. modifcaRicorrenzaDataTerminazion(frequenza, ripetizione, eventoBase)

ALT

ok

9b.1a.1 L'evento non è ricorrente

Contratti Gestire gli Eventi

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

1) statoCucina()

Interrogazione al sistema: non ha post-condizioni.
Nemmeno pre-condizioni, potrebbe essere fatta in qualsiasi momento.

1a.1) annullaEvento(eventoSingolo: Evento, applicaSuccessivi: si/no)

Pre-condizioni: Non ha pre-condizioni, potrebbe essere fatta in qualsiasi momento.
Post-condizioni:

- [Se eventoSingolo.stato = "pianificato" o "in corso"]:
  - eventoSingolo.stato = "annullato"
  - [per ogni istanza *serv* di Servizio che eventoSingolo **da luogo**]
    - *serv*.stato = "annullato"
- [Se applicaSuccessivi = "si"]:
  - [se eventoSingolo **occorre** in un'istanza *er* di EventoRicorrente]:
    - applica le modifiche precedenti a tutte le istanze *eventoS* di EventoSingolo che **occorrono** in *er* e con *eventoS*.dataInizio > eventoSingolo.dataInizio

1b.1) eliminaEvento(eventoSingolo: Evento, applicaSuccessivi: si/no)

Pre-condizioni: Non ha pre-condizioni, potrebbe essere fatta in qualsiasi momento.
Post-condizioni: Se *org* **organizza** eventoSingolo

- [per ogni istanza *serv* di Servizio che *evento* **da luogo**]
  - [per ogni istanza *adc* di AttivitàDiCucina che l'istanza *foglio* di FoglioDiRiepilogo **organizza**, dove *serv* **ha associato** *foglio*]
    - [per ogni istanza *cdc di CompitoDiCucina in cui **viene svolta** adc*]
      - l'istanza *cdc* è stata eliminata
    - l'istanza *adc* è stata eliminata
  - l'istanza *serv* è stata eliminata
- [Se applicaSuccessivi = "si"]:
  - [se eventoSingolo **occorre** in un'istanza *er* di EventoRicorrente]:
    - Sono state applicate le modifiche precedenti a tutte le istanze *eventoS di* EventoSingolo che **occorrono** in *er* e con *eventoS*.dataInizio > eventoSingolo.dataInizio
    - l'istanza *eventoS* è stata eliminata
- l'istanza eventoSingolo è stata eliminata

1c.1) terminaEvento(eventoSingolo: Evento)

Pre-condizioni: Non ha pre-condizioni, potrebbe essere fatta in qualsiasi momento.

Post-condizioni:

[Se eventoSingolo.stato = “in corso”]:

- eventoSingolo.stato = “terminato”
- [per ogni istanza *serv* di Servizio che eventoSingolo **da luogo**]
  - *serv*.stato = “terminato”

1c.2) aggiungiNoteFinali(noteFinali: testo, documentiFinali?: documenti)

Pre-condizioni:

- è in corso la modifica di un EventoSingolo *evento*
- *evento*.stato = “terminato”

Post-condizioni:

- *evento*.noteFinali = noteFinali
- [Se documentiFinali è definito]:
  - *evento*.dcumentiFinali = documentiFinali

1d.1) aggiungiRimanenza(adc: AttivitàDiCucina, rimanenza: Rimanenza)

Pre-condizioni: Non ha pre-condizioni, potrebbe essere fatta in qualsiasi momento.

Post-condizioni: *adc* **produce** rimanenza

(1-2)a.1) apriEvento(evento: Evento)

Pre-condizioni: evento è in stato “pianificato” o “in corso”

Post-condizioni: - -

(1-2)a.2) modificaEvento(dataInizio?:Data, nPartecipanti?: numero, location?:Location, chef?: Chef)

Pre-condizioni:

- è in corso la modifica di un EventoSingolo *evento*

Post-condizioni:

- [Se dataInzio è definita]:
  - *evento*.dataInizio = dataInizio
- [Se nPartecipanti è definita]:
  - *evento*.nPartecipanti = nPartecipanti
- [Se location è definita]:
  - [Se esiste un’istanza di Location *loc* tale che *evento* **si svolge in** *loc*]:
    - è stata eliminata l’associazione **si svolge in** tra *evento* e *loc*

        - *evento* **si svolge in** location
    - [Se chef è definito]:
        - [Se esiste un'istanza di Chef *chef* tale che *evento* **ha associato** *chef*]:
            - è stata eliminata l'associazione **ha associato** tra *evento* e *chef*
        - *evento* **ha associato** chef

2) creaSchedaEvento(cliente: testo, dataInizio: Data, nPartecipanti: numero, location: Location)

Pre-condizioni: -

Post-condizioni:

- è stata creata l'istanza *evento* di EventoSingolo
- *evento*.cliente = cliente
- *evento*.dataInizio = dataInizio
- *evento*.nPartecipanti = nPartecipanti
- *evento*.noteParticolari = EMPTY
- *evento*.stato = "pianificato"
- *evento*.noteFinali = EMPTY
- *evento*.documentiFinali = EMPTY
- *org* **è responsabile di** *evento*
- *evento* **si svolge in** location

3) aggiungiServizio(tipologia: testo, giornoInizio: numero, oraInizio: ora, oraFine: ora, luogoSvolgimento?: testo, applicaSuccessivi: si/no)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*

Post-condizioni: Se il giorno della data *evento*.dataInizio + giornoInizio non è passato:

- è stata creata l'istanza *servizio* di Servizio
- *servizio*.tipologia = tipologia
- *servizio*.giornoInizio = giornoInizio
- *servizio*.oraInizio = oraInizio
- *servizio*.oraFine = oraFine
- [Se luogoSvolgimento è definito]:
    - *servizio*.luogoSvolgimento = luogoSvolgimento
- *servizio*.stato = "da confermare"
- *evento* **da luogo a** *servizio*
- [Se applicaSuccessivi = "si"]:
    - [Se *evento* **occorre** in un'istanza *er* di EventoRicorrente]:
        Per ogni istanza *eventoS* di EventoSingolo che **occorre** in *er* e per cui *eventoS*.dataInizio > *evento*.dataInizio
        - crea un servizio *sCopia* copia di *servizio*

- *eventoS* **da luogo a** *sCopy*

3a.1)modificaServizio(servizio: Servizio, tipologia?: testo, giornoInizio?: numero, oraInizio?: ora, oraFine?: ora, luogoSvolgimento?: testo, applicaSuccessivi: si/no)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*
- *evento* **da luogo a** servizio

Post-condizioni: Se giornoInizio è definito e il giorno della data *evento*.dataInizio + giornoInizio non è passato:

- [Se tiplogia è definito]
  - servizio.tipologia = tipologia
- [Se giornoInizio è definito]
  - servizio.giornoInizio = giornoInizio
- [Se oraInizio è definito]
  - servizio.giornoInizio = giornoInizio
- [Se oraInizio è definito]
  - servizio.oraInizio = oraInizio
- [Se oraFine è definito]
  - servizio.oraFine = oraFine
- [Se luogoSvolgimento è definito]
  - servizio.luogoSvolgimento = luogoSvolgimento
- [Se applicaSuccessivi = “si”]:
  - [Se *evento* **occorre** in un’istanza *er* di EventoRicorrente]:

    Per ogni istanza *eventoS* di EventoSingolo che **occorre** in *er* e per cui *eventoS*.dataInizio > *evento*.dataInizio
    - [Se esiste un servizio sCopy copia di *servizio*] applica le modifiche precedenti a sCopy

3a.1a.1)annullaServizio(servizio: Servizio, applicaSuccessivi: si/no)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *eventoBase*
- *eventoBase* **da luogo a** servizio

Post-condizioni:

- servizio.stato = “annullato”
- [Se applicaSuccessivi = “si”]:
  - [Se esiste l’associazione **occorre** tra eventoSingolo e un’istanza *er* di EventoRicorrente]:

- applica le modifiche precedenti a tutte le istanze *eventoS* di EventoSingolo che **occorrono** con *er* e con *eventoS*.dataInizio > eventoSingolo.dataInizio

3a.1b.1)eliminaServizio(servizio: Servizio, applicaSuccessivi: si/no)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *eventoBase*
- *eventoBase* **da luogo a** servizio

Post-condizioni:

- L'associazione **da luogo a** tra servizio ed *eventoBase* è eliminata
- L'associazione **assegnato a** tra servizio e un'istanza Menu *menu* è stata eliminata
- L'istanza servizio è stata eliminata
- [Se applicaSuccessivi = "si"]:
  - [Se esiste l'associazione **occorre** tra eventoSingolo e un'istanza *er* di EventoRicorrente]:
    - applica le modifiche precedenti a tutte le istanze *eventoS* di EventoSingolo che **occorrono** con *er* e con *eventoS*.dataInizio > eventoSingolo.dataInizio

4) aggiungiNoteParticolari(note: testo)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*

Post-condizioni:

- *evento*.noteParticolari = note

5) associaChef(chef: Chef, applicaSuccessivi:si/no)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*

Post-condizioni:

- L'associazione **ha associato** tra *evento* e chef viene creata
- [Se *evento* **occorre** in un'istanza *er* di EventoRicorrente e applicaSuccessivi ha valore *si*]:
  [Per ogni EventoSingolo *eventoS* che **occorre** in *er* dove *eventoS*.dataInizio > *evento*.dataInizio]
  - *eventoS* **ha associato** chef

5a.1)modifcaChefAssegnato(chef: Chef, applicaSuccessivi:si/no)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*
- *evento* **ha associato** un'istanza *chefPrec* di Chef

Post-condizioni:

- L'associazione **ha associato** tra *evento* e *chefPrec* è stata eliminata
- *evento* **ha associato** chef
- [Se *evento* **occorre** in un'istanza *er* di EventoRicorrente e applicaSuccessivi ha valore *si*]:
  [Per ogni EventoSingolo *eventoS* che **occorre** in *er* dove *eventoS*.dataInizio > *evento*.dataInizio]
  - [se e*ventoS* **ha associato** un'istanza *chefPrec_* di Chef]
    L'associazione **ha associato** tra *eventoS* e *chefPrec_* è stata eliminata
  - *eventoS* **ha associato** chef

6) confermaMenu(servizio: Servizio)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*
- *evento* **da luogo a** servizio

Post-condizioni:

- servizio.stato = "confermato"
- [Se per ogni istanza *s* di Servizio a cui *evento* **da luogo**, *s*.stato vale "confermato"]:
  - *evento*.stato = "in corso"

6a.1) proponiModifica(servizio: Servizio, voce: Voce, operazione: Testo)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*
- *evento* **da luogo a** servizio

Post-condizioni:

- [Se operazione è "aggiunta"] :
  - è stata creata un'istanza *proposta* di PropostaAggiunta
  - servizio **riceve** *proposta*
  - *proposta* **include** voce

- [Altrimenti se operazione è “eliminazione” e a servizio viene **assegnato** un’istanza *menu* di Menu e voce **appartiene** a *menu* oppure voce **appartiene** a un’istanza *sezione* di Sezione tale che *menu* **contiene** *sezione*] :
    - è stata creata un’istanza *proposta* di PropostaEliminazione
    - servizio **riceve** *proposta*
    - *proposta* **scarta** voce

7) generaCompitoDiServizio(servizio: Servizio, membroPersonale: PersonaleDiServzio, ts: TurnoDiServizio, ruolo?:testo)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*
- *evento* **da luogo a** servizio

Post-condizioni: Se membroPersonale è **disponibile per** ts:

- è stata creata l’istanza *compitoDiServizio* di CompitoDiServizio
- servizio **prevede** *compitoDiServizio*
- *compitoDiServizio* **si tiene in** ts
- membroPersonale **svolge** *compitoDiServizio*
- [Se ruolo è definito] :
    - *compitoDiServizio*.ruolo = ruolo

7a.1) liberaMembroPersonale(cds: CompitoDiServizio, membroPersonale: PersonaleDiServzio)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*
- *evento* **da luogo a** un’istanza *servizio* di Servizio
- *servizio* **prevede** cds
- cds **è svolto da** membroPersonale

Post-condizioni:

- è stata eliminata l’associazione **svolge** tra membroPersonale e cds

7b.1) AssegnaMembroPersonale(membroPersonale: PersonaleDiServzio, cds: CompitoDiServizio)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*
- *evento* **da luogo a** servizio
- servizio **prevede** cds

Post-condizioni:

- cds **è svolto da** membroPersonale

7c.1) impostaRuoloCompitoDiServizio(cds: CompitoDi Servizio, ruolo:testo)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*
- *evento* **da luogo a** servizio
- servizio **prevede** cds

Post-condizioni:

- cds.ruolo = ruolo

8) richiediAssistenzaCuoco(servizio: Servizio, cuoco: Cuoco, turnoDiServizio: TurnoDiServizio)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*
- *evento* **da luogo a** servizio

Post-condizioni:

- è stata creata l'istanza di Assistenza *assistenza*
- *assistenza* **si tiene in** turnoDiServizio
- servizio **richiede** *assistenza*
- cuoco **presta** *assistenza*

8a.1) liberaCuoco(servizio: Servizio, cuoco: Cuoco)

Pre-condizioni:

- è in corso la modifica di un EventoSingolo *evento*
- *evento* **da luogo a** servizio
- servizio **richiede** *assistenza*
- cuoco **presta** *assistenza*

Post-condizioni:

- è stata eliminata l'associazione **richiede** tra servizio e *assistenza*
- è stata eliminata l'istanza *assistenza*

9) setEventoRicorrente(frequenza :Frequenza, ripetizioni: numero)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *eventoBase*
- *eventoBase* **si svolge in** un'istanza *location* di Location

Post-condizioni:

- L'istanza *eventoRicorrente di* EventoRicorrente è stata creata
- *eventoBase* **occorre** in *eventoRicorrente* è stata creata
- *eventoRicorrente*.frequenza = frequenza
- Ripeti per *i* che va da 1 a (ripetizioni - 1) , i++

- L'istanza *eventoS* di EventoSingolo è stata creata
- *eventoS* **occorre** in *eventoRicorrente*
- *eventoS*.cliente = *eventoBase.*cliente
- *eventoS*.dataInizio = *eventoBase.*dataInizio +(frequenza * *i*)
- *eventoS*.nPartecipanti = *eventoBase.*npartecipanti
- *eventoS*.stato = "pianificato"
- *eventoS* **si svolge in** *location*
- [se *eventoBase* **ha associato** un'istanza *chef* di Chef] *eventoS* **ha associato** *chef*
- Per ogni istanza *servizio* di Servizio a cui *eventoBase* **da luogo**
  - L'istanza *servizioCopia* di Servizio è stata creata
  - i campi di *servizioCopia* vengono valorizzati come quelli di *servizio*
  - *servizioCopia*.stato = "da confermare"
  - *eventoS* **da luogo a** *servizioCopia*
  - Per ogni istanza *proposta* di PropostaDiAggiunta che *servizio* **riceve** tale che *proposta*.accettata è "sì"
    - L'istanza *propostaCopia* di PropostaDiAggiunta è stata creata
    - *propostaCopia*.accettata = no
    - *propostaCopia* **include** l'istanza *voce* di Voce, tale che *proposta* **include** *voce*
    - *servizioCopia* **riceve** *propostaCopia*
  - Per ogni istanza *proposta* di PropostaDiEliminazione che *servizio* **riceve** tale che *proposta*.accettata è "sì"
    - L'istanza *propostaCopia* di PropostaDiEliminazione è stata creata
    - *propostaCopia*.accettata = no
    - *propostaCopia* **scarta** l'istanza *voce* di Voce, tale che *proposta* **scarta** *voce*
    - *servizioCopia* **riceve** *propostaCopia*

9a.1)setEventoRicorrenteData(frequenza :Frequenza, dataTerminazione: numero)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *eventoBase*
- *eventoBase* **si svolge in** un'istanza *location* di Location

Post-condizioni:

- L'istanza *eventoRicorrente di* EventoRicorrente è stata creata
- *eventoBase* **occorre** in *eventoRicorrente* è stata creata
- *eventoRicorrente*.frequenza = frequenza
- Ripeti per *i* che va da *eventoBase.*dataInizio+frequenza fino a quando i >= dataTerminazione, i+= frequenza

- L'istanza *eventoS* di EventoSingolo è stata creata
- *eventoS* **occorre** in *eventoRicorrente*
- *eventoS*.cliente = *eventoBase.*cliente
- *eventoS*.dataInizio = *eventoBase*.dataInizio + i
- *eventoS*.nPartecipanti = *eventoBase.*npartecipanti
- *eventoS*.stato = "pianificato"
- *eventoS* **si svolge in** *location*
- [se *eventoBase* **ha associato** un'istanza *chef* di Chef] *eventoS* **ha associato** *chef*
- Per ogni istanza *servizio* di Servizio a cui *eventoBase* **da luogo**:
    - L'istanza *servizioCopia* di Servizio è stata creata
    - i campi di *servizioCopia* vengono valorizzati come quelli di *servizio*
    - *servizioCopia*.stato = "da confermare"
    - *eventoS* **da luogo a** *servizioCopia*
    - Per ogni istanza *proposta* di PropostaDiAggiunta che *servizio* **riceve** tale che *proposta*.accettata è "sì":
        - L'istanza *propostaCopia* di PropostaDiAggiunta è stata creata
        - *propostaCopia*.accettata = no
        - *propostaCopia* **include** l'istanza *voce* di Voce, tale che *proposta* **include** *voce*
        - *servizioCopia* **riceve** *propostaCopia*
    - Per ogni istanza *proposta* di PropostaDiEliminazione che *servizio* **riceve** tale che *proposta*.accettata è "sì":
        - L'istanza *propostaCopia* di PropostaDiEliminazione è stata creata
        - *propostaCopia*.accettata = no
        - *propostaCopia* **scarta** l'istanza *voce* di Voce, tale che *proposta* **scarta** *voce*
        - *servizioCopia* **riceve** *propostaCopia*

9b.1)modifcaRicorrenzaRipetizioni(frequenza :Frequenza, ripetizioni: numero, eventoBase: EventoSingolo)

Pre-condizioni:

- è in corso la definizione di un EventoSingolo *evento*

Post-condizioni: Se *evento* **occorre** in un'istanza *er* di EventoRicorrente e eventoBase **occorre** in *er* :

- Per ogni EventoSingolo *esDaEliminare* che **occorre** in *er* con *esDaEliminare.*dataInizio > *evento.*dataInizio :
    - l'associazione **si svolge in** tra un'istanza *location* di Location ed *esDaEliminare* è stata eliminata

        - l'istanza *esDaEliminare* è stata eliminata
- *er.*numeroRipetizioni = ripetizioni
- *er.*frequenza = frequenza
- Ripeti per *i* che va da 1 a (ripetizioni - 1) :
    - L'istanza *eventoS* di EventoSingolo è stata creata
    - *eventoS* **occorre** in *er*
    - *eventoS*.cliente = eventoBase.cliente
    - *eventoS*.dataInizio = eventoBase.dataInizio +(frequenza * *i*)
    - *eventoS*.nPartecipanti = eventoBase.nPartecipanti
    - *eventoS*.stato = "pianificato"
    - *eventoS* **si svolge in** nell'istanza *location* di Location tale che eventoBase **si svolge in** *location*
    - [se eventoBase **ha associato** un'istanza *chef* di Chef] *eventoS* **ha associato** *chef*
    - Per ogni istanza *servizio* di Servizio a cui eventoBase **da luogo**:
        - L'istanza *servizioCopia* di Servizio è stata creata
        - i campi di *servizioCopia* vengono valorizzati come quelli di *servizio*
        - *servizioCopia*.stato = "da confermare"
        - *eventoS* **da luogo a** *servizioCopia*
        - Per ogni istanza *proposta* di PropostaDiAggiunta che *servizio* **riceve** tale che *proposta*.accettata è "sì":
            - L'istanza *propostaCopia* di PropostaDiAggiunta è stata creata
            - *propostaCopia*.accettata = no
            - *propostaCopia* **include** l'istanza *voce* di Voce, tale che *proposta* **include** *voce*
            - *servizioCopia* **riceve** *propostaCopia*
        - Per ogni istanza *proposta* di PropostaDiEliminazione che *servizio* **riceve** tale che *proposta*.accettata è "sì":
            - L'istanza *propostaCopia* di PropostaDiEliminazione è stata creata
            - *propostaCopia*.accettata = no
            - *propostaCopia* **scarta** l'istanza *voce* di Voce, tale che *proposta* **scarta** *voce*
            - *servizioCopia* **riceve** *propostaCopia*

9c.1) modifcaRicorrenzaDataTerminazione(frequenza :Frequenza, dataTerminazione: data, eventoBase: EventoSingolo)

Pre-condizioni:

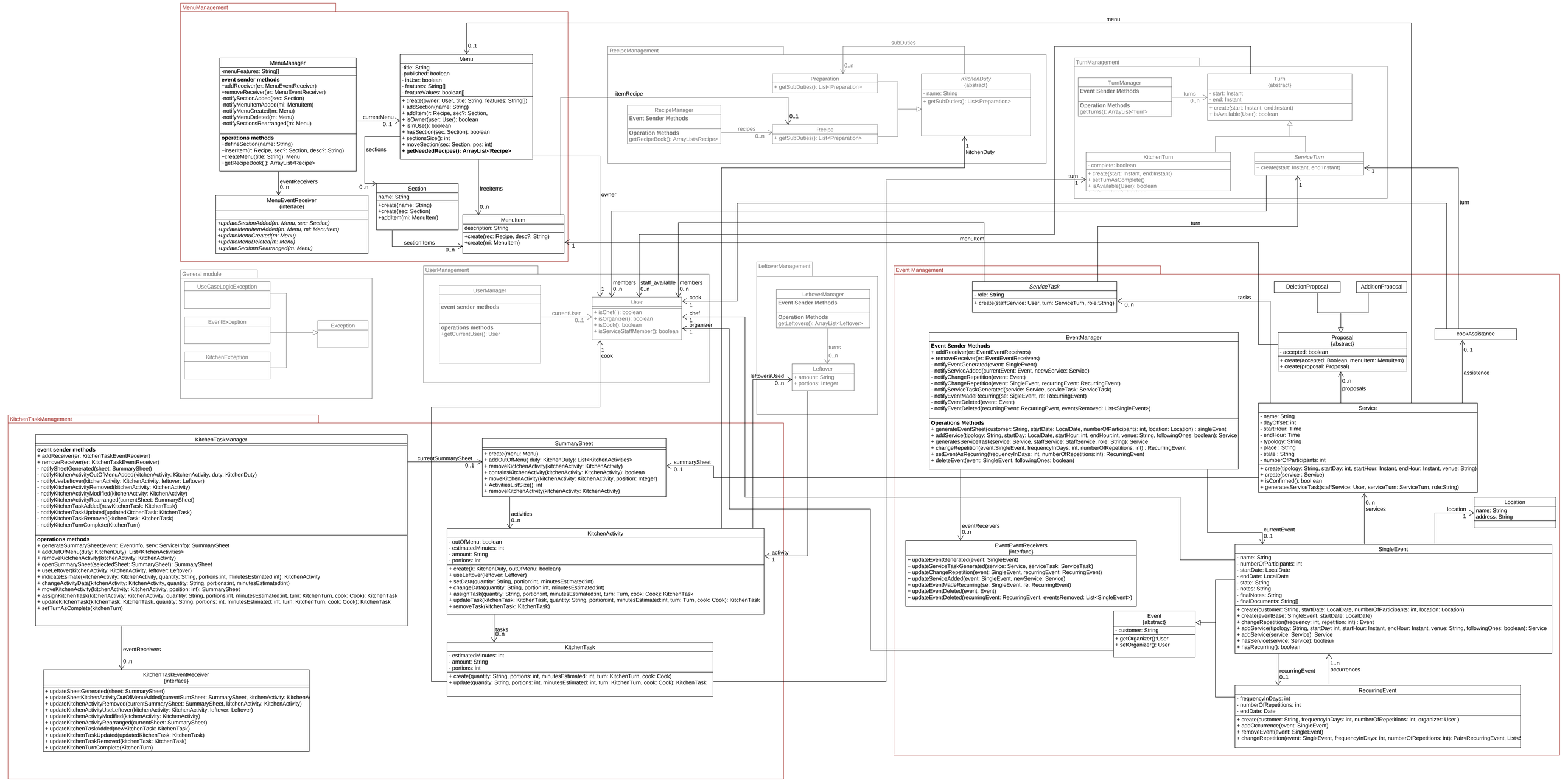
- è in corso la definizione di un EventoSingolo *evento*

Post-condizioni: Se *evento* **occorre** in un'istanza *er* di EventoRicorrente e eventoBase **occorre** in *er* :

- Per ogni EventoSingolo *esDaEliminare* che **occorre** in *er* con *esDaEliminare.*dataInizio > *evento.*dataInizio :
    - l'associazione **si svolge in** tra un'istanza *location* di Location ed *esDaEliminare* è stata eliminata
    - [per ogni istanza *serv* di Servizio a cui *esDaEliminare* **da luogo**]
        - [per ogni istanza *adc* di AttivitàDiCucina che l'istanza *foglio* di FoglioDiRiepilogo **organizza**, dove *serv* **ha associato** *foglio*]
            - [per ogni istanza *cdc di CompitoDiCucina in cui **viene svolta** adc*]
                - l'istanza *cdc* è stata eliminata
            - l'istanza *adc* è stata eliminata
        - l'istanza *serv* è stata eliminata
    - l'istanza *esDaEliminare è stata eliminata*
- *er.*frequenza = frequenza
- *er.*dataTerminazione = dataTerminazione
- Ripeti per *i* che va da eventoBase.dataInizio+frequenza fino a quando i >= dataTerminazione, i+= frequenza :
    - L'istanza *eventoS* di EventoSingolo è stata creata
    - *eventoS* **occorre** in *er*
    - *eventoS*.cliente = eventoBase.cliente
    - *eventoS*.dataInizio = eventoBase.dataInizio + i
    - *eventoS*.nPartecipanti = eventoBase.nPartecipanti
    - *eventoS*.stato = "pianificato"
    - *eventoS* **si svolge in** nell'istanza *location* di Location tale che eventoBase **si svolge in** *location*
    - [se eventoBase **ha associato** un'istanza *chef* di Chef] *eventoS* **ha associato** *chef*
    - Per ogni istanza *servizio* di Servizio a cui eventoBase **da luogo**:
        - L'istanza *servizioCopia* di Servizio è stata creata
        - i campi di *servizioCopia* vengono valorizzati come quelli di *servizio*
        - *servizioCopia*.stato = "da confermare"
        - *eventoS* **da luogo a** *servizioCopia*
        - Per ogni istanza *proposta* di PropostaDiAggiunta che *servizio* **riceve** tale che *proposta*.accettata è "sì":
            - L'istanza *propostaCopia* di PropostaDiAggiunta è stata creata
            - *propostaCopia*.accettata = no
            - *propostaCopia* **include** l'istanza *voce* di Voce, tale che *proposta* **include** *voce*
            - *servizioCopia* **riceve** *propostaCopia*

- Per ogni istanza *proposta* di PropostaDiEliminazione che *servizio* **riceve** tale che *proposta*.accettata è “sì”:
    - L’istanza *propostaCopia* di PropostaDiEliminazione è stata creata
    - *propostaCopia*.accettata = no
    - *propostaCopia* **scarta** l’istanza *voce* di Voce, tale che *proposta* **scarta** *voce*
    - *servizioCopia* **riceve** *propostaCopia*

Design Class Diagram (DCD)



UI

eventManager:
EventManager

userManager:
UserManager

er:
EventEventReceivers

generateEvent(customer, startDate, numberOfParticipants, location)

getCurrentUser( )

user

ALT
[NOT user.isOrganizer()]

throw UseCaseLogicException

[startDate.compareTo
(LocalDate.now()) < 0]

throw EventException

create(customer, startDate, numberOfParticipants,location, user)

event:
SingleEvent

this.customer = customer

this.startDate = startDate

this.numberOfParticipants = numberOfParticipants

this.state = "planned"

setLocation(location)

setOrganizer(user)

create()

services:
List<Service>

event

this.currentEvent = event

notifyEventGenerated(currentEvent)

for each
er in receivers

updateEventGenerated(currentEvent)

event

UI
eventManager: EventManager
userManager: UserManager
EventManager.currentEvent: SingleEvent
er: EventEventReceiver
addService(tipology, startDay, startHour, endHour, venue, followingOnes)
getCurrentUser( )
user
ALT
[eventManager.currentEvent == null]
throw UseCaseLogicException
[NOT this.currentEvent.getOrganizer() == user
OR (currentEvent.startDate.getDay() + startDay)
.compareTo(LocalDate.now()) < 0]
throw EventException
addService(tipology, startDay, startHour, endHour, venue, followingOnes)
create(tipology, startDay, startHour, endHour, venue)
newService: Service
this.tipology = tipology
this.startDay = startDay
this.startHour = startHour
OPT
[venue != null]
this.venue = venue
this.endHour = endHour
this.confirmed = false
create()
proposals: List<Proposal>
newService
Crea il nuovo servizio per l'evento corrente
currentEvent.services: List<Service>
add(newService)
newService
notifyServiceAdded(currentEvent, newService)
for each er in receivers
updateServiceAdded(currentEvent, newService)
OPT
[followingOnes AND currentEvent.hasRecurring()]
for each occurrence in currentEvent.getRecurringEvent().occurrences
occurrence: SingleEvent
OPT
[occurrence.startDate >= currentEvent.startDate]
addService(newService)
create(newService)
sCopy: Service
this.tipology = newService.tipology
this.startDay = newService.startDay
this.startHour = newService.startHour
OPT
[newService.venue != null]
this.venue = newService.venue
this.endHour = newService.endHour
this.confirmed = false
create()
proposals: List<Proposal>
sCopy
Copia il servizio creato per tutte le altre occorrenze di evento successive
occurrence.services: List<Service>
add(sCopy)
newSCopy
notifyServiceAdded(occurrence, newService)
for each er in receivers
updateServiceAdded(occurrence, newSCopy)
newService

UI
eventManager: EventManager
userManager: UserManager
eventBase.recurringEvent(): RecurringEvent
currentEvent.getRecurringEvent().occurences: List<SingleEvent>
er: EventEventReceiver

changeRepetition(eventBase, frequency, repetitions)
getCurrentUser( )
user

ALT
[eventManager.currentEvent == null]
throw UseCaseLogicException

[NOT this.currentEvent.getOrganizer() == user
OR this.currentEvent.hasRecurring()
AND this.currentEvent.getRecurringEvent()
.getOccurrences().contains(eventBase))]
throw EventException

changeRepetition(currentEvent,frequency, repetitions)
create()
eventsRemoved: List<SingleEvent>

LOOP
[for each event
in this.getOccurences()]
OPT
[event.startDate > currentEvent.startDate]
add(event)
remove(event)

Note..
Rimozione delle future occorrenze di evento

this.numberOfRepetitions = repetitions
this.frequencyInDays = frequency

LOOP
[for each i
from i = 1 to i = (repetition-1)]
create(eventBase, eventBase.startDate + (frequency, i))
newEvent: SingleEvent
this.numberOfParticipants = event.numberOfParticipants
this.client = event.client
this.startDate = startDate
setLocation(eventBase.getLocation())
this.state = "planned"
create(s)
services: List<Service>

LOOP
for each s
in newEvent.services
create(s)
sCopy: Service
this.dayOffset = s.dayOffset
this.startHour = s.startHour
this.endHour = s.endHour
this.typology = s.typology
OPT
[sCopy.place != null]
this.place = sCopy.place
this.state = "to be confirmed"
create()
proposals: List<Proposal>
for each p
in s.proposals
ALT
[p.accepted]
create(p)
pCopy: Proposal
this.accepted = no
setMenuItem(p.getMenuItem)
pCopy
add(pCopy)
sCopy
add(sCopy)

newEvent
add(newEvent)

this, eventsRemoved
notifyEventDeleted(recurringEvent, eventsRemoved)
for each
er in receivers
updateEventDeleted(recurringEvent, eventsRemoved)
notifyChangeRepetition(currentEvent, recurringEvent)
for each
er in receivers
updateChangeRepetition(currentEvent, recurringEvent)
recurringEvent

UI

eventManager: EventManager

userManager: UserManager

event.getRecurringEvent(): RecurringEvent

event.getRecurringEvent().occurences: List<SingleEvent>

er: EventEventReceivers

deleteEvent(event, followingOnes)

getCurrentUser( )

user

ALT
[NOT user.isOrganiser()]

throw UseCaseLogicException

[NOT event.getOrganizer() == user]

throw EventException

OPT
[event.hasRecurring()]

removeEvent(event)

remove(event)

Elimina l'evento singolo

notifyEventDeleted(event)

for each
er in receivers

updateEventDeleted(event)

Elimina gli eventi successivi

OPT
[followingOnes == true AND event.hasRecurring()]

for each
e in event.getRecurringEvent().occurences

OPT
[e.startDate.compareTo(event.startDate) > 0]

removeEvent(e)

remove(e)

notifyEventDeleted(event)

for each
er in receivers

updateEventDeleted(event)

Elimina la ricorrenza

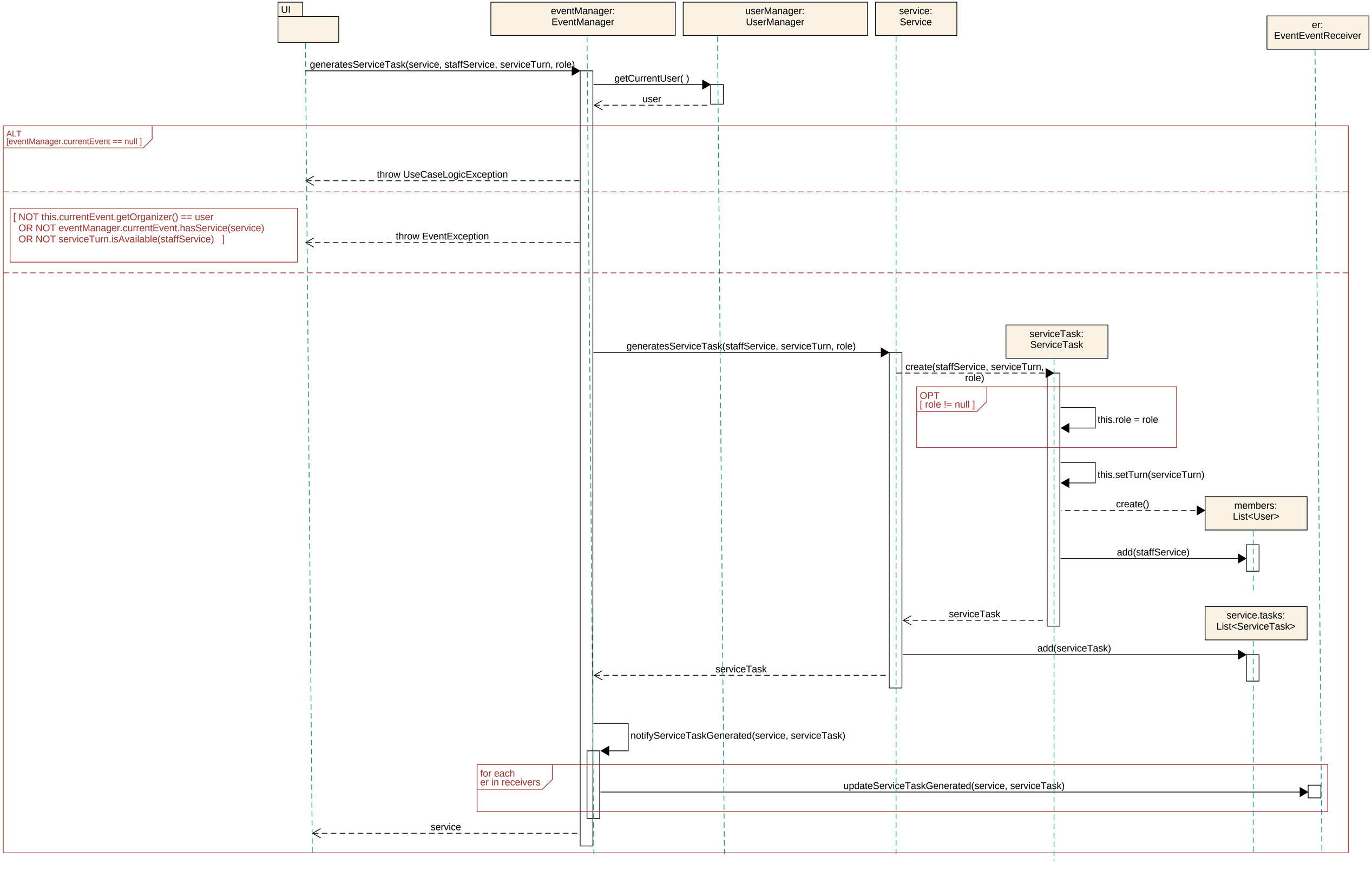
OPT
[event.getRecurringEvent().occurences.size() == 0]

notifyEventDeleted(event.getRecurringEvent())

for each
er in receivers

updateEventDeleted(event.getRecurringEvent())

UI
eventManager: EventManager
userManager: UserManager
er: EventEventReceivers
setEventAsRecurring(frequencyInDays, numberOfRepetitions)
getCurrentUser( )
user
ALT
[NOT user.isOrganizer() OR this.currentEvent == null]
throw UseCaseLogicException
[NOT this.currentEvent .getOrganizer() == user]
throw EventException
create(this.currenEvent.customer, frequencyInDays, numberOfRepetitions, user)
event: RecurringEvent
this.customer = customer
this.frequencyInDays = frequencyInDays
this.numberOfRepetitions = numberOfRepetitions
setOrganiser(user)
create()
occurences: List<SingleEvent>
recurringEvent
addOccurrence(currentEvent)
add(event)
LOOP
for i = 1
to i = numberOfRepetitions - 1
i+
create(this.currentEvent, currentEvent.startDate + frequencyInDays * i)
event: SingleEvent
this.customer = event.customer
this.startDate = calculatedDate
this.numberOfParticipants = event.numberOfParticipants
this.state = "planned"
setLocation(event.location)
setLocation(event.getLocation())
OPT
[event.getChef() != null]
setChef(chef)
create()
services: List<Service>
for each s
in event.services
create(s)
sCopy: Service
this.dayOffset = s.dayOffset
this.startHour = s.startHour
this.endHour = s.endHour
this.typology = s.typology
OPT
[sCopy.place != null]
this.place = sCopy.place
this.state = "to be confirmed"
create()
proposals: List<Proposal>
for each p
in s.proposals
OPT
[p.accepted]
create(p)
pCopy: Proposal
this.accepted = false
setMenuItem(p.getMenuItem)
pCopy
add(pCopy)
sCopy
add(sCopy)
event
setOrganizer(user)
addOccurrence(event)
add(event)
notifyEventMadeRecurring(currentEvent, recurringEvent)
for each
er in receivers
updateEventMadeRecurring(currentEvent, recurringEvent)
recurringEvent


UI
eventManager:
EventManager
userManager:
UserManager
service:
Service
er:
EventEventReceiver
generatesServiceTask(service, staffService, serviceTurn, role)
getCurrentUser( )
user
ALT
[eventManager.currentEvent == null]
throw UseCaseLogicException
[NOT this.currentEvent.getOrganizer() == user
OR NOT eventManager.currentEvent.hasService(service)
OR NOT serviceTurn.isAvailable(staffService)]
throw EventException
generatesServiceTask(staffService, serviceTurn, role)
serviceTask:
ServiceTask
create(staffService, serviceTurn,
role)
OPT
[role != null]
this.role = role
this.setTurn(serviceTurn)
create()
members:
List<User>
add(staffService)
serviceTask
service.tasks:
List<ServiceTask>
add(serviceTask)
serviceTask
notifyServiceTaskGenerated(service, serviceTask)
for each
er in receivers
updateServiceTaskGenerated(service, serviceTask)
service