ANNO ACCADEMICO 2023/2024

Sviluppo Applicazioni Software

Gestire i Turni

Luca Barra, Jasmine Bizzo

UNIVERSITÀ
DI TORINO

DIPARTIMENTO DI INFORMATICA

1
Gestire i turni

Informazioni generali

Nome Caso d'Uso: Gestire i turni

Portata: Sistema

Livello: Obiettivo utente

Attore primario: Organizzatore

Parti interessate: Chef, Cuochi

Pre-condizioni: L'attore deve essere identificato come Organizzatore

Garanzie di successo o post-condizioni: I turni di cucina e di servizio sono stati impostati

Scenario principale di successo

#	Attore	Sistema
1	Aggiunge un turno di cucina e, opzionalmente, segna un luogo	Registra il turno della cucina.
	Ripete il passo 1 finché non è soddisfatto	
2	Opzionalmente, controlla gli eventi.	Visualizza gli eventi.
	Se non vuole lavorare su nessun evento termina il Caso d'Uso	
3	Sceglie un evento su cui lavorare.	Visualizza l'evento selezionato.
4	Aggiunge un turno di servizio, impostandone un orario.	Registra l'orario del turno di servizio.
5	Opzionalmente, aggiunge una scadenza oltre la quale non si può togliere la disponibilità.	Registra la scadenza.
6	Opzionalmente, cambia l'orario del turno di servizio.	Registra le modifiche.
	Se vuole lavorare su altri servizi torna al punto 4	

	Se vuole lavorare su altri eventi torna al passo 3, altrimenti termina il Caso d'Uso	

Estensione 1a

#	Attore	Sistema
1a.1	Modifica un turno di cucina.	Registra la modifica del turno di cucina.

Eccezione 1a

#	Attore	Sistema
1a.1	Aggiunge un turno di cucina e, opzionalmente, segna un luogo.	Non si hanno i permessi necessari.
	Termina il Caso d'Uso	

Eccezione 1b

#	Attore	Sistema
1b.1	Aggiunge un turno di cucina e, opzionalmente, segna un luogo.	Esiste già un turno in quell'orario.
	Termina il Caso d'Uso	

Eccezione 1a.1a

#	Attore	Sistema
1a.1a.1	Modifica un turno di cucina.	Non si hanno i permessi necessari.
	Termina il Caso d'Uso	

Estensione 1b

#	Attore	Sistema

1b.1	Cancella un turno di cucina.	Elimina il turno di cucina specificato.

Eccezione 1b.1a

#	Attore	Sistema
1b.1a.1	Cancella un turno di cucina.	Non si hanno i permessi necessari.
	Termina il Caso d'Uso	

Estensione 1c

#	Attore	Sistema
1c.1	Modifica tutti i turni di cucina in un determinato giorno, settimana.	Registra le modifiche di tutti i turni di cucina aventi determinate caratteristiche.

Eccezione 1c.1a

#	Attore	Sistema
1c.1a.1	Modifica tutti i turni di cucina in un determinato giorno, settimana.	Non si hanno i permessi necessari.
	Termina il Caso d'Uso	

Estensione 1d

#	Attore	Sistema
1d.1	Elimina tutti i turni di cucina in un determinato giorno, settimana.	Registra l'eliminazione di tutti i turni di cucina aventi determinate caratteristiche.

Eccezione 1d.1a

#	Attore	Sistema

1d.1a.1	Elimina tutti i turni di cucina in un determinato giorno, settimana.	Non si hanno i permessi necessari.
	Termina il Caso d'Uso	

Estensione 5a

#	Attore	Sistema
5a.1	Rimuove la scadenza.	Toglie la scadenza oltre la quale non si può più rimuovere la disponibilità.

Eccezione 5a.1a

#	Attore	Sistema
5a.1a.1	Rimuove la scadenza.	Il turno non ha una scadenza impostata.
	Torna al passo 5	

Estensione 5b

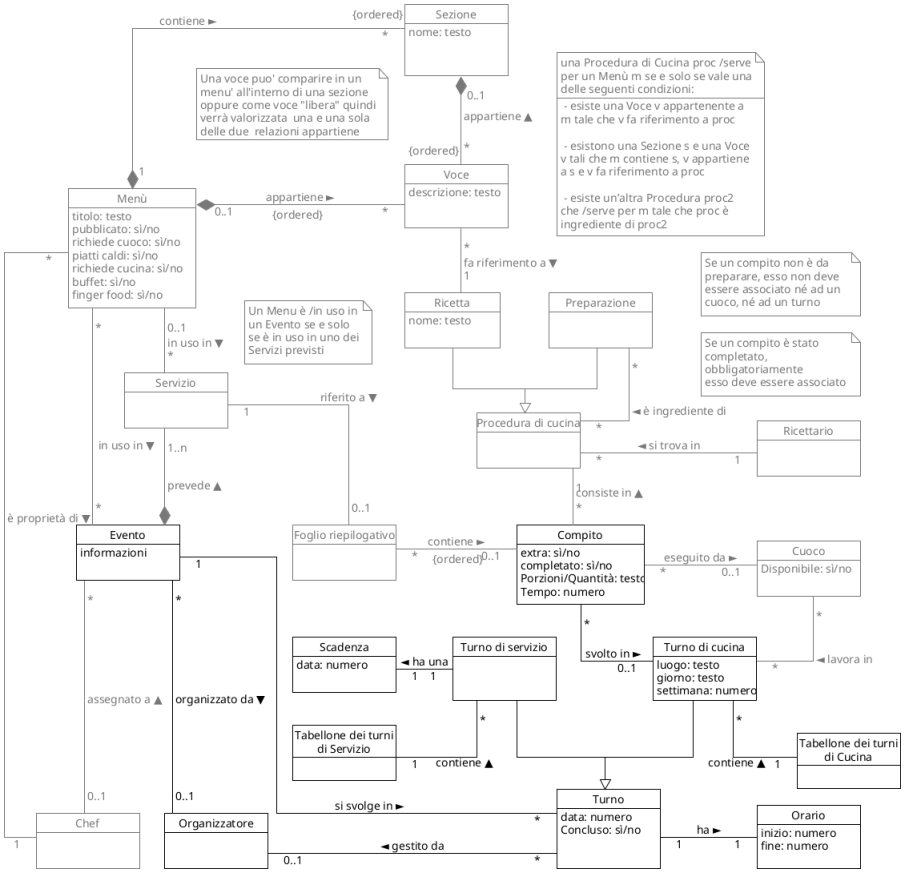
#	Attore	Sistema
5b.1	Modifica la scadenza.	Registra la nuova scadenza.

Estensione 6a

#	Attore	Sistema
6a.1	Cancella un turno di servizio.	Registra la cancellazione del turno di servizio..

2

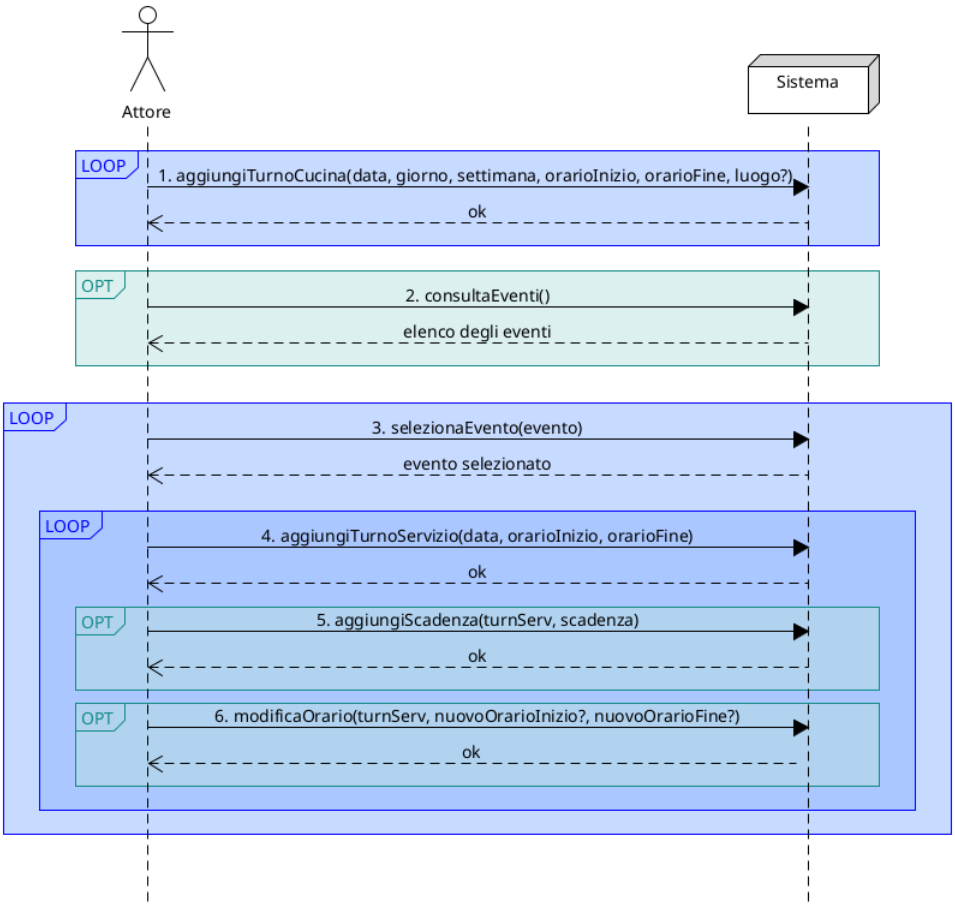
Modello di Dominio



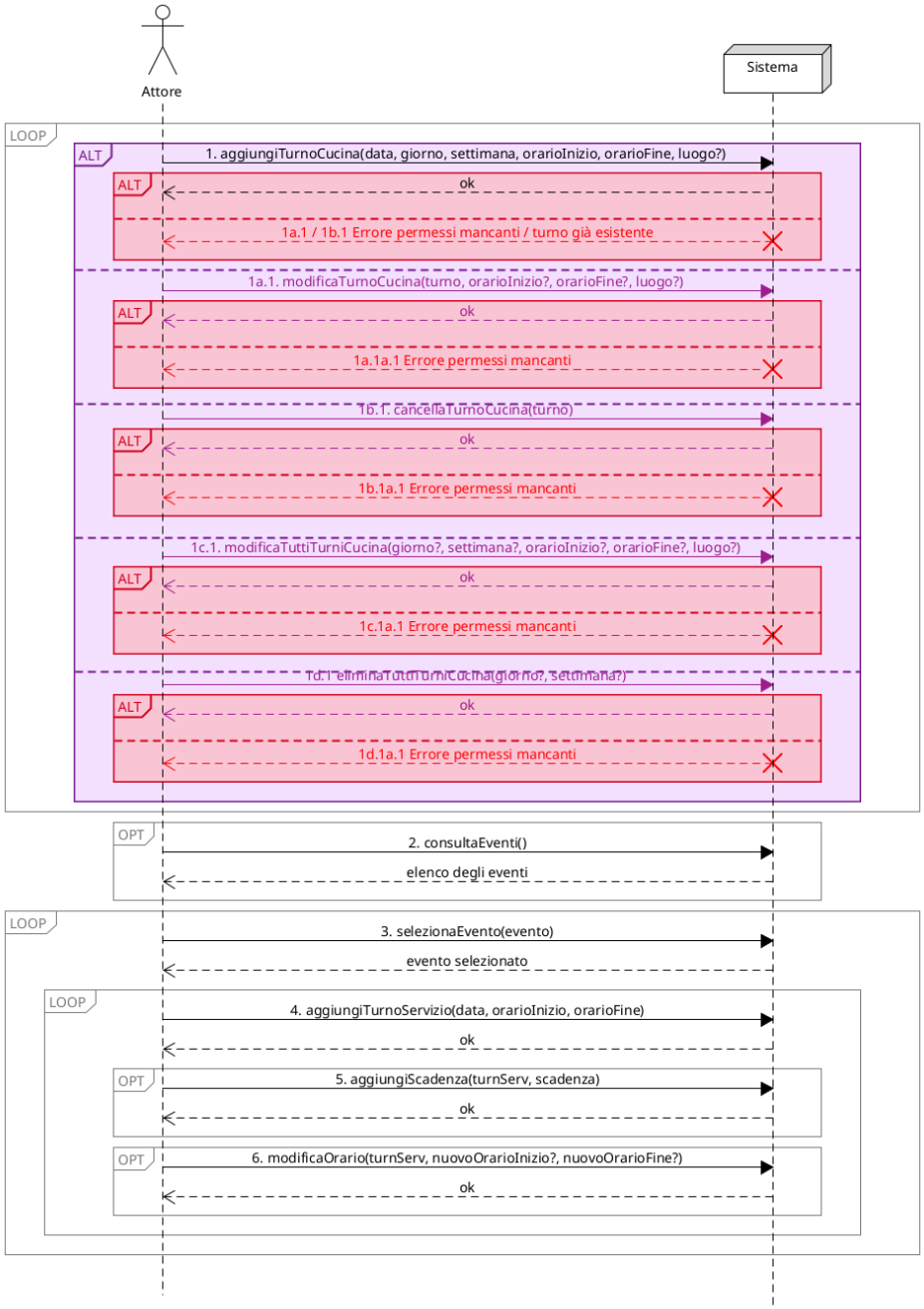
3

SSD - System Sequence Diagram

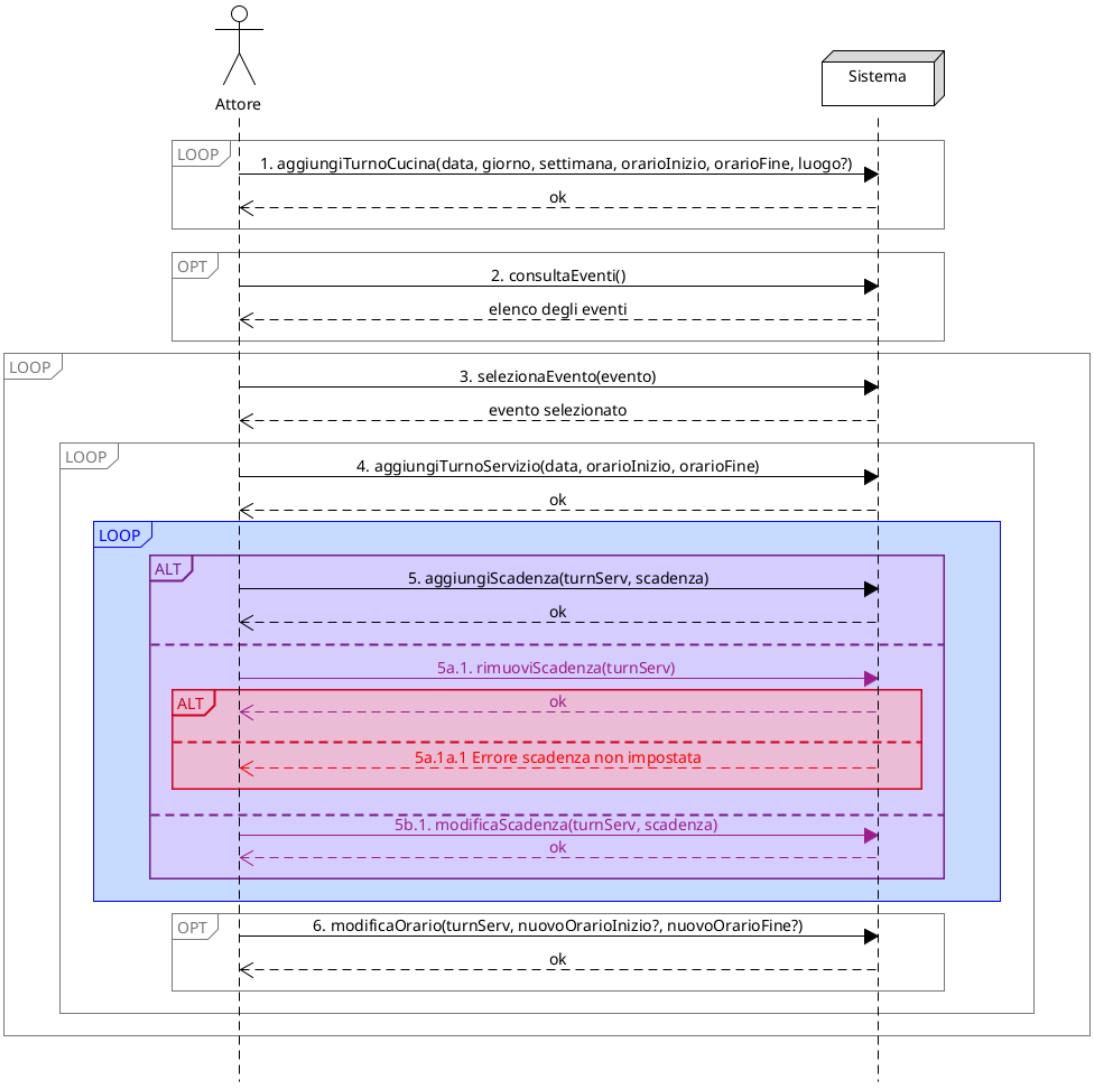
3.1 Scenario principale



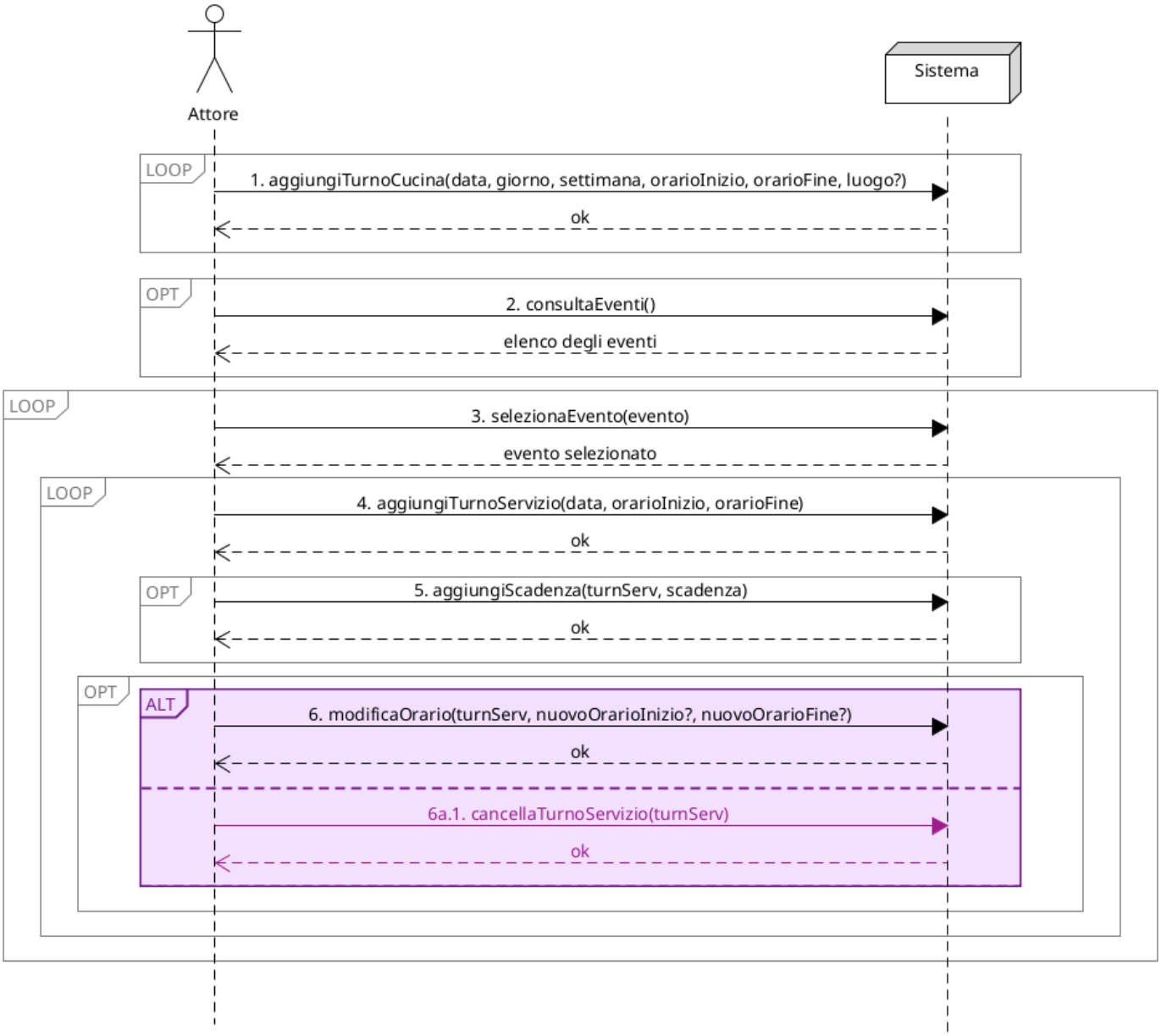
3.2 Eccezioni ed estensioni - Passo 1



3.3 Eccezioni ed estensioni - Passo 5



3.4 Eccezioni ed estensioni - Passo 6



4
Contratti

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

4.1 Passo 1

1. aggiungiTurnoCucina(data: numero, giorno: testo, settimana: numero, orarioInizio: numero, orarioFine: numero, luogo?: Testo):

Pre-condizioni:

- è in corso la modifica di *ttc* TabelloneTurniCucina.

Post-condizioni: Se *tc* è **gestito da** *org* e non esiste un altro turno con data == *tc.data*, orarioInizio >= *tc.orario*.orarioInizio e orarioFine <= *tc.orario*.orarioFine

- è stata creata un'istanza *tc* di TurnoCucina;
- è stata creata un'istanza *o* di Orario;
- è stata creata un'associazione **ha** *tc* e *o*;
- è stata creata un'associazione **contiene** *tc* e *ttc*;
- *tc*.data = *data*;
- *tc*.giorno = *giorno*;
- *tc*.settimana = *settimana*;
- [se *luogo* è specificato] *turno*.luogo = *luogo*;
- *tc*.concluso = no;
- *o*.inizio = *orarioInizio*;
- *o*.fine = *orarioFine*.

1a.1 modificaTurnoCucina(turno: TurnoCucina, orarioInizio?: numero, orarioFine?: numero, luogo?: Testo):

Pre-condizioni:

- è in corso la modifica di *ttc* TabelloneTurnoCucina;
- *ttc* **contiene** *tc*;
- [se *orarioInizio* o *orarioFine* è specificato] non esiste un *tc* tale che *tc*.orario.inizio > *orario* e *tc*.orario.fine < *orario*.

Post-condizioni: Se *tc* è **gestito da** *org*:

- [se *orarioInizio* è specificato] *turno*.orario.inizio = *orarioInizio*;
- [se *orarioFine* è specificato] *turno*.orario.fine = *orarioFine*;
- [se *luogo* è specificato] *turno*.luogo = *luogo*.

1b.1 cancellaTurnoCucina(turno: TurnoCucina):

Pre-condizioni:

- è in corso la modifica di *ttc* TabelloneTurniCucina;
- *ttc* **contiene** *tc*.

Post-condizioni: Se *tc* è **gestito da** *org*

- ⟨ per ogni Compito *c* **presente in** turno ⟩:
  - l'associazione **svolto in** tra *c* e turno è stata eliminata.
- l'associazione **contiene** tra *tc* e *ttc* è stata eliminata;
- l'associazione **gestito da** tra *tc* e *org* è stata eliminata;
- l'associazione **si svolge in** tra *tc* e *Evento* è stata eliminata;
- l'associazione **ha** tra *tc* e *o* è stata eliminata;
- l'istanza *o* di Orario cessa di esistere;
- l'istanza *tc* di TurnoCucina cessa di esistere.

1c.1 modificaTuttiTurniCucina(giorno?: Testo, settimana?: numero, orarioInizio?: numero, orarioFine?: numero, luogo?: Testo):

Pre-condizioni:

- è in corso la modifica di *ttc* TabelloneTurniCucina;
- *ttc* **contiene** *tc*.

Post-condizioni: Se *tc* è **gestito da** *org*

- [se *giorno* e *settimana* sono specificati] ⟨per ogni TurnoCucina *tc* presente in *ttc* con *tc*.data.giorno == *giorno* e *tc*.data.settimana == *settimana* ⟩:
  - [se *orarioInizio* è specificato] *tc*.orario.inizio = *orarioInizio*;
  - [se *orarioFine* è specificato] *tc*.orario.fine = *orarioFine*;
  - [se *luogo* è specificato] *tc*.luogo = *luogo*.
- [se *giorno* è specificato] ⟨per ogni TurnoCucina *tc* presente in *ttc* con *tc*.data.giorno == *giorno* ⟩:
  - [se *orarioInizio* è specificato] *tc*.orario.inizio = *orarioInizio*;
  - [se *orarioFine* è specificato] *tc*.orario.fine = *orarioFine*;
  - [se *luogo* è specificato] *tc*.luogo = *luogo*.
- [se *settimana* è specificato] ⟨per ogni TurnoCucina *tc* presente in *ttc* con *tc*.data.settimana == *settimana* ⟩:
  - [se *orarioInizio* è specificato] *tc*.orario.inizio = *orarioInizio*;
  - [se *orarioFine* è specificato] *tc*.orario.fine = *orarioFine*;
  - [se *luogo* è specificato] *tc*.luogo = *luogo*.

1d.1 eliminaTuttiTurniCucina(giorno: Testo, settimana: Testo):

Pre-condizioni:

- è in corso la modifica di *ttc* TabelloneTurniCucina;
- *ttc* **contiene** *tc*.

Post-condizioni:

- [se *giorno* e *settimana* sono specificati] ⟨per ogni TurnoCucina *tc* presente in *ttc* con *tc*.data.giorno == *giorno* e *tc*.data.settimana == *settimana* ⟩:
  - ⟨ per ogni Compito *c* **presente in** *t* ⟩:
    - l'associazione tra *c* e *tc* è stata eliminata.
  - l'associazione **contiene** tra *tc* e *ttc* è stata eliminata;
  - l'associazione **gestito da** tra *tc* e *org* è stata eliminata;
  - l'associazione **si svolge in** tra *tc* e *Evento* è stata eliminata;
  - l'associazione **ha** tra *tc* e *o* è stata eliminata;
  - l'istanza *o* di Orario cessa di esistere;
  - l'istanza *tc* di TurnoCucina cessa di esistere.
- [se *giorno* è specificato] ⟨per ogni TurnoCucina *tc* presente in *ttc* con *tc*.data.giorno == *giorno* ⟩:
  - ⟨ per ogni Compito *c* **presente in** *t* ⟩:
    - l'associazione tra *c* e *tc* è stata eliminata.
  - l'associazione **contiene** tra *tc* e *ttc* è stata eliminata;
  - l'associazione **gestito da** tra *tc* e *org* è stata eliminata;
  - l'associazione **si svolge in** tra *tc* e *Evento* è stata eliminata;
  - l'associazione **ha** tra *tc* e *o* è stata eliminata;
  - l'istanza *o* di Orario cessa di esistere;

  - l'istanza *tc* di TurnoCucina cessa di esistere.

- [se *settimana* è specificata] ⟨per ogni TurnoCucina *tc* presente in *ttc* con *tc*.data.settimana == *settimana* ⟩:
  - ⟨ per ogni Compito *c* **presente in** *t* ⟩:
    - l'associazione tra *c* e *tc* è stata eliminata.
  - l'associazione **contiene** tra *tc* e *ttc* è stata eliminata;
  - l'associazione **gestito da** tra *tc* e *org* è stata eliminata;
  - l'associazione **si svolge in** tra *tc* e *Evento* è stata eliminata;
  - l'associazione **ha** tra *tc* e *o* è stata eliminata;
  - l'istanza *o* di Orario cessa di esistere;
  - l'istanza *tc* di TurnoCucina cessa di esistere.

4.2 Passo 2

2. consultaEventi():

Pre-condizioni:

Post-condizioni:

Note:-

consultaEventi() è un'interrogazione di sistema quindi non ha pre e post condizioni.

4.3 Passo 3

3. selezionaEvento(evento: Evento):

Pre-condizioni:

- *evento* è **organizzato da** *org*.

Post-condizioni:

4.4 Passo 4

4. aggiungiTurnoServizio(data: numero, orarioInizio: numero, orarioFine: numero):

Pre-condizioni:

- è in corso la modifica di *tts* TabelloneTurniServizio;
- è in corso la definizione dei turni di servizio in *e* Evento e *servizio* **è previsto** da *e*.

Post-condizioni:

- è stata creata un'istanza *ts* di TurnoServizio;
- è stata creata un'istanza *o* di Orario;
- è stata creata un'associazione **ha** *ts* e *o*;
- è stata creata un'associazione **contiene** tra *ts* e *tts*;
- *ts*.data = *data*;
- *o*.inizio = *orarioInizio*;
- *o*.fine = *orarioFine*.
- *ts*.concluso = no.

4.5 Passo 5

5. aggiungiScadenza(turnoServizio: TurnoServizio, scadenza: numero):

Pre-condizioni:

- è in corso la modifica di *tts* TabelloneTurniServizio;
- *tts* **contiene** *ts*;
- *ts* è **gestito da** *org*;
- *ts*.concluso = no.

Post-condizioni:

- è stata creata un'istanza *s* di Scadenza;
- *s*.data = *scadenza*;
- è stata creata l'associazione **ha una** tra *ts* e *s*.

5a.1 rimuoviScadenza(turnoServizio: TurnoServizio):

Pre-condizioni:

- è in corso la modifica di *tts* TabelloneTurniServizio;
- *tts* **contiene** *ts*;
- *ts* è **gestito da** *org*;
- *ts*.concluso = no.

Post-condizioni: Se esiste l'associazione **ha una** tra *ts* e un'istanza *s* di Scadenza:

- l'associazione **ha una** tra *ts* e *s* è stata eliminata;
- l'istanza *s* di Scadenza cessa di esistere.

5b.1 modificaScadenza(turnoServizio: TurnoServizio, scadenza: numero):

Pre-condizioni:

- è in corso la modifica di *tts* TabelloneTurniServizio;
- *tts* **contiene** *ts*;
- *ts* è **gestito da** *org*;
- *ts*.concluso = no.

Post-condizioni:

- *s*.data = *scadenza*.

4.6 Passo 6

6. modificaOrario(turnoServizio: TurnoServizio, nuovoOrarioInizio?: numero, nuovoOrarioFine?: numero,):

Pre-condizioni:

- è in corso la modifica di *tts* TabelloneTurniServizio;
- *tts* **contiene** *ts*;
- *ts* è **gestito da** *org*;
- *ts*.concluso = no.

Post-condizioni:

- [se *orarioInizio* è specificato] *ts*.Orario.inizio = *orarioInizio*;
- [se *orarioFine* è specificato] *ts*.Orario.fine = *orarioFine*;

6a.1 cancellaTurnoServizio(turnoServizio: TurnoServizio):

Pre-condizioni:

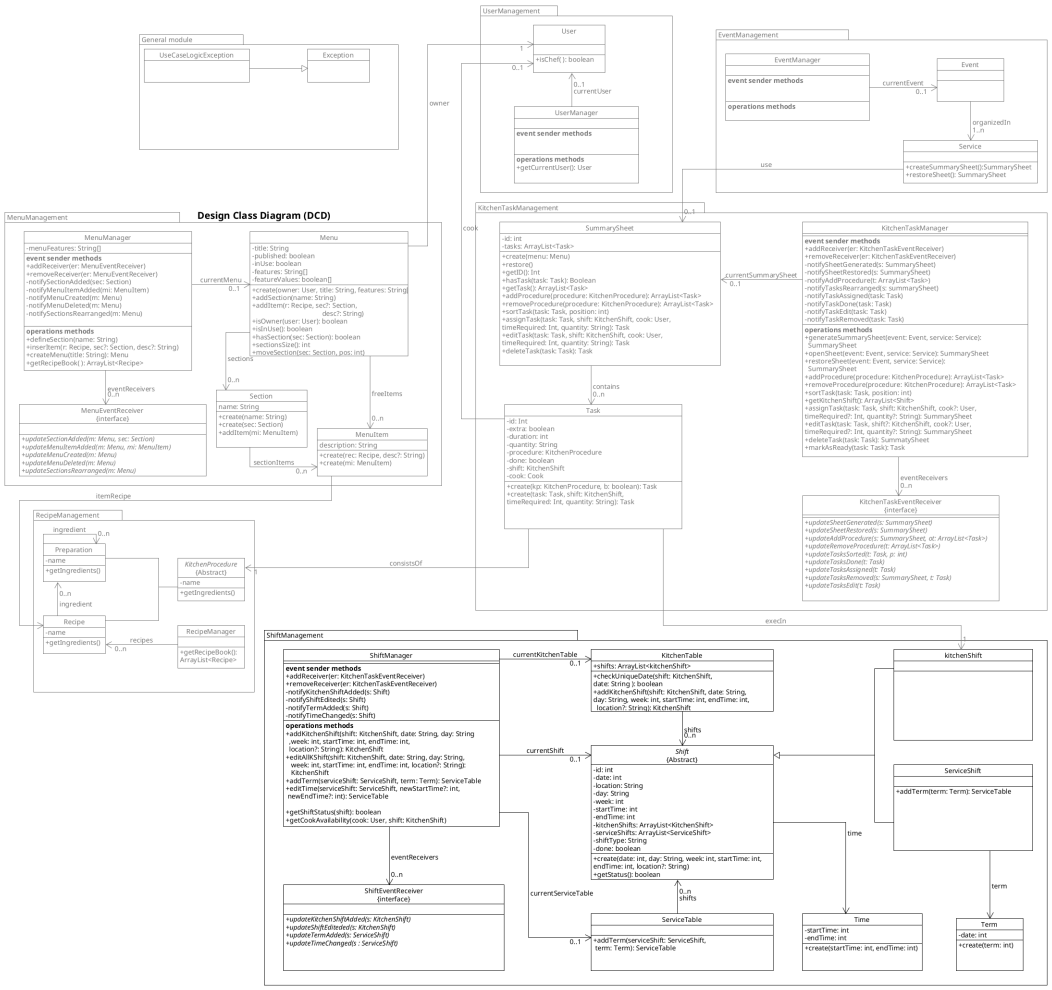
- è in corso la modifica di *tts* TabelloneTurniServizio;
- *tts* **contiene** *ts*;
- *ts* è **gestito da** *org*;
- *ts*.concluso = no.

Post-condizioni:

- [se esiste l'associazione **ha una** tra *ts* e un'istanza *s* di Scadenza]:
  - questa è stata eliminata;
  - l'istanza *s* di Scadenza cessa di esistere.
- l'associazione **contiene** tra *ts* e *tts* è stata eliminata;
- l'associazione **gestito da** tra *ts* e *org* è stata eliminata;
- l'associazione **si svolge in** tra *ts* e *Evento* è stata eliminata;
- l'associazione **ha** tra *ts* e *o* è stata eliminata;
- l'istanza *o* di Orario cessa di esistere;
- l'istanza *ts* di TurnoServizio cessa di esistere.

5

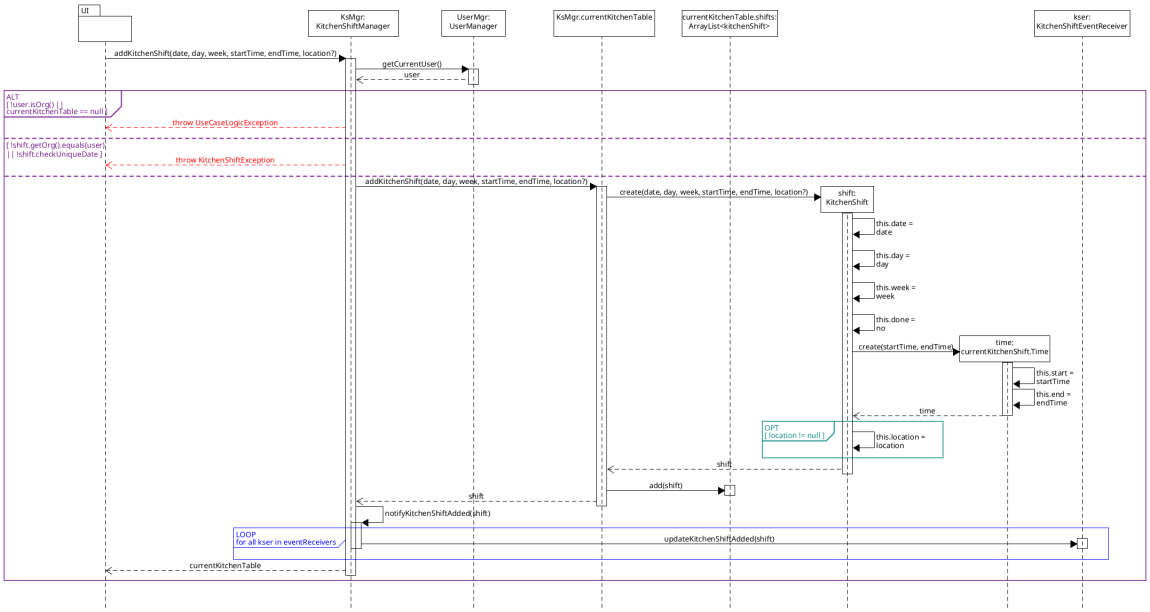
DCD - Design Class Diagram

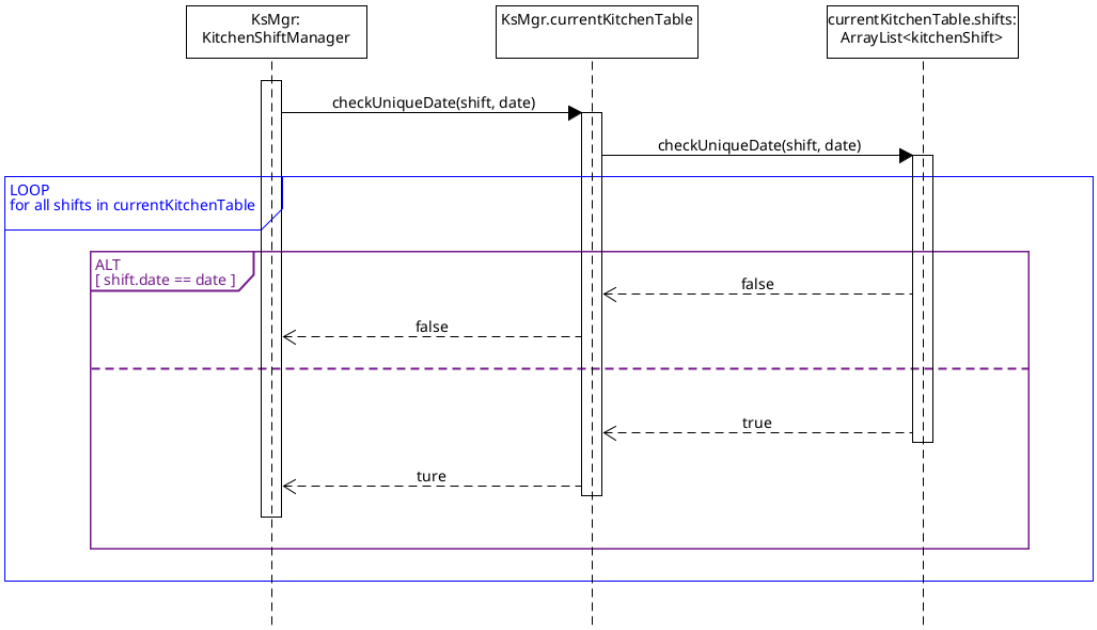


6

DSD - Design Sequence Diagram

1. addKitchenShift(shift, date, day, week, startTime, endTime, location?)

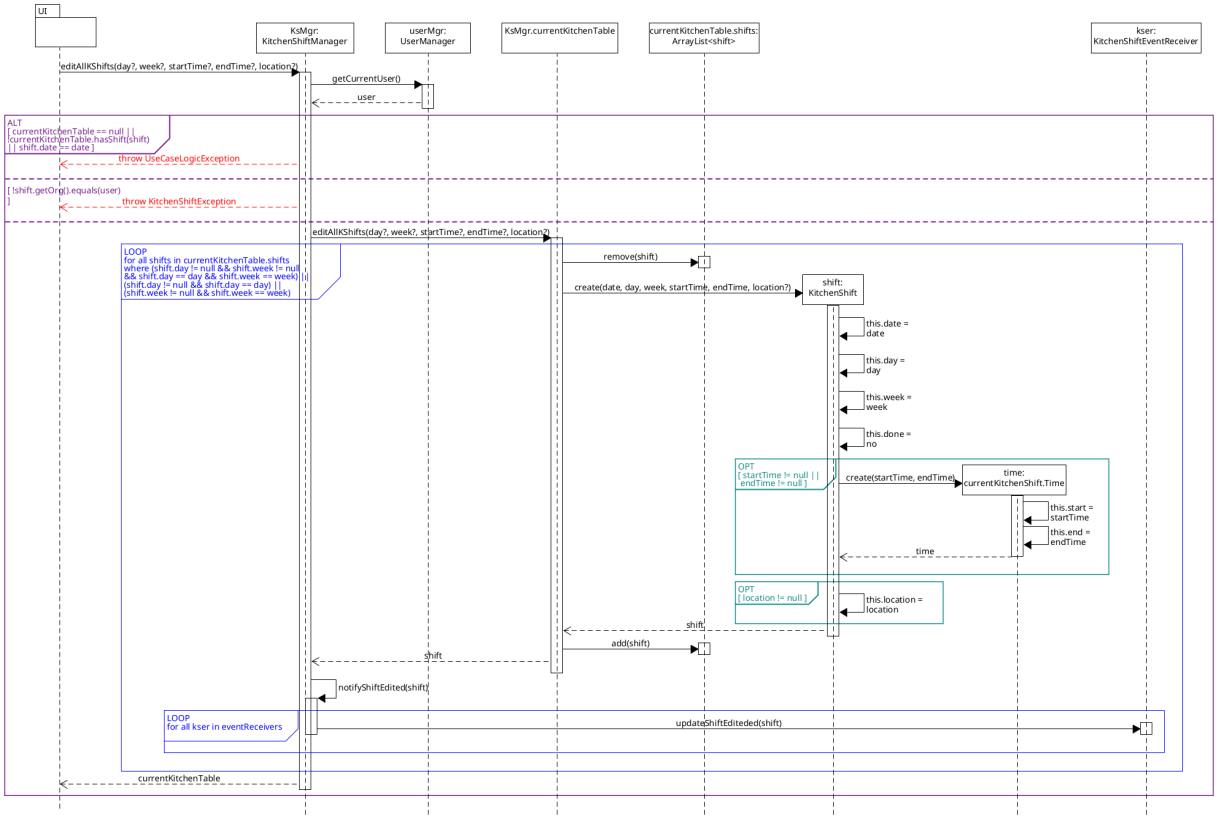




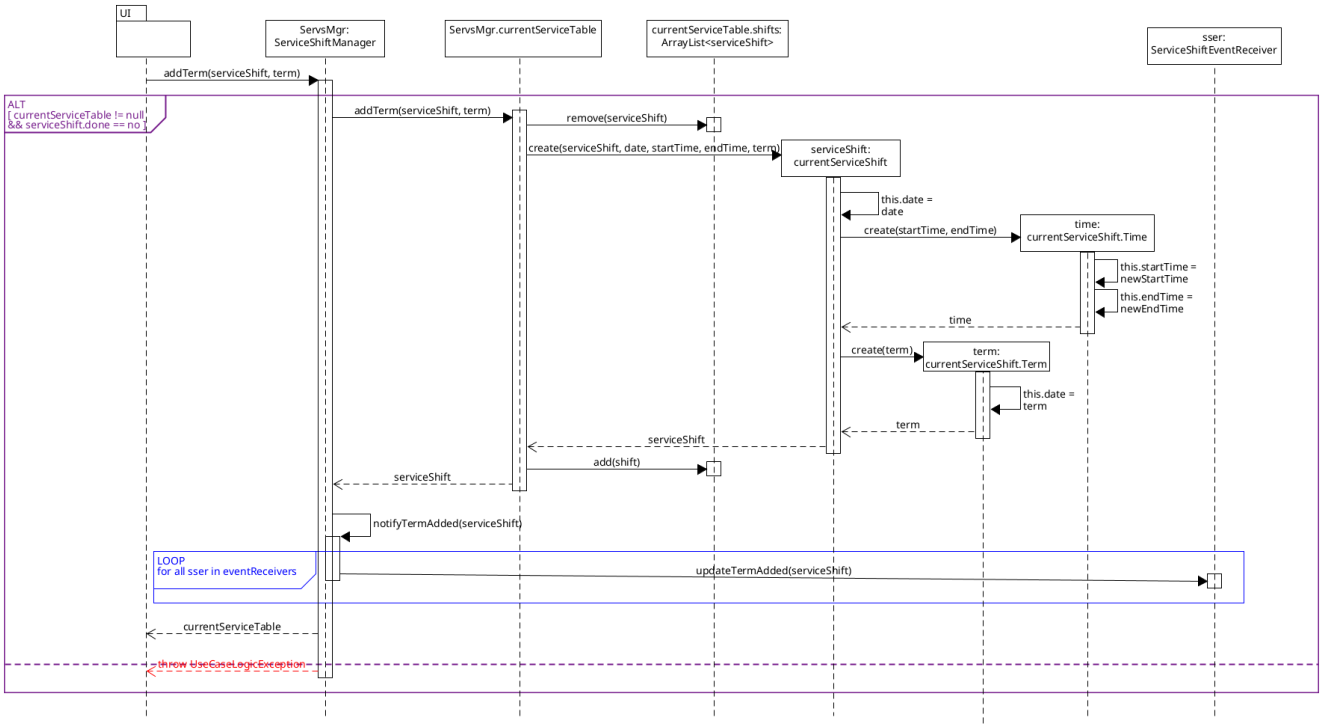
Note:-

DSD per illustrare l'operazione per controllare che due turni non si sovrappongano.

1c. editAllShifts(day?, week?, startTime?, endTime?, location?)



5. addTerm(serviceShift, term)



6. editTime(serviceShift, newTime)