# Sviluppo delle Applicazioni Software Laboratorio a.a 2023/24

Lorenzo Bonincontro, Alberto Aiello

# Gestione dei compiti della cucina

# Informazioni generali

Nome caso d'uso: Gestione dei compiti della Cucina

Portata: Sistema

#### Livello: Obiettivo Utente

Attore primario: Chef Parti Interessate: Cuoco

Pre-condizioni: L'utente deve essersi autenticato come Chef ed esiste almeno un evento in corso

affidatogli.

**Garanzie di successo o post-condizioni**: I compiti assegnati sono stati registrati e riportati correttamente sia nel tabellone dei turni che nel foglio riepilogativo.

## Scenario principale di successo

#	Attore	Sistema
1.	Genera il <mark>foglio riepilogativo</mark> per un <mark>servizio</mark> di <mark>evento (</mark> di cui ha ricevuto l'incarico)	Precompila il foglio riepilogativo per il servizio in modo che contenga tutte le preparazioni e ricette previste dal menù associato al servizio.
2.	Opzionalmente, crea un <mark>compito</mark> specificando <mark>ricetta/preparazione</mark> e lo aggiunge all'elenco dei compiti.	Registra il compito e lo aggiunge alla lista dei compiti.
3.	Opzionalmente ordina l'elenco	Aggiorna il foglio riepilogativo con il nuovo ordinamento
	Ripete dal passo 2 finchè non è soddisfatto. oppure Altrimenti continua o termina il caso d'uso.	
4.	Opzionalmente consulta il <mark>tabellone dei</mark>	Fornisce il tabellone dei turni

	turni	
5.	Opzionalmente Assegna un <mark>compito</mark> specificando cosa ( <mark>ricetta/preparazione</mark> ), quando ( <mark>turno</mark> ) e opzionalmente chi ( <mark>cuoco</mark> ).	Registra l'assegnazione
6.	Opzionalmente indica una stima del tempo richiesto per lo svolgimento del compito appena assegnato, e la /porzioni preparate in un dato assegnamento	Registra le informazioni fornite sul tabellone dei turni e sul foglio riepilogativo.
	Ripete dal passo 2 finchè non soddisfatto	
	Termina il caso d'uso	

## Estensione 1a

#	Attore	Sistema
1a.1	Apre un foglio riepilogativo già esistente per modificarlo	Fornisce il foglio riepilogativo già esistente desiderato
	Continua dal passo 2	

# Eccezione 1a.1a

#	Attore	Sistema
1a.1a.1	Apre un foglio riepilogativo già esistente per modificarlo	Il foglio riepilogativo non è di proprietà dell'attore che sta cercando di modificarlo, pertanto non si può proseguire.
	Termina il caso d'uso	

# **Estensione 1b**

#	Attore	Sistema
1b.1	Elimina foglio riepilogativo	Elimina il Foglio riepilogativo del servizio
	Termina il caso d'uso	

# Eccezione 1b.1a

#	Attore	Sistema
1b.1a.1	Elimina foglio riepilogativo	Il foglio riepilogativo non è di proprietà dell'attore che sta cercando di eliminarlo, pertanto non si può proseguire.
	Termina il caso d'uso	

## **Estensione 5a**

#	Attore	Sistema
5a.1	Modifica compito	Registra modifica
	Termina il caso d'uso	

# **Estensione 5b**

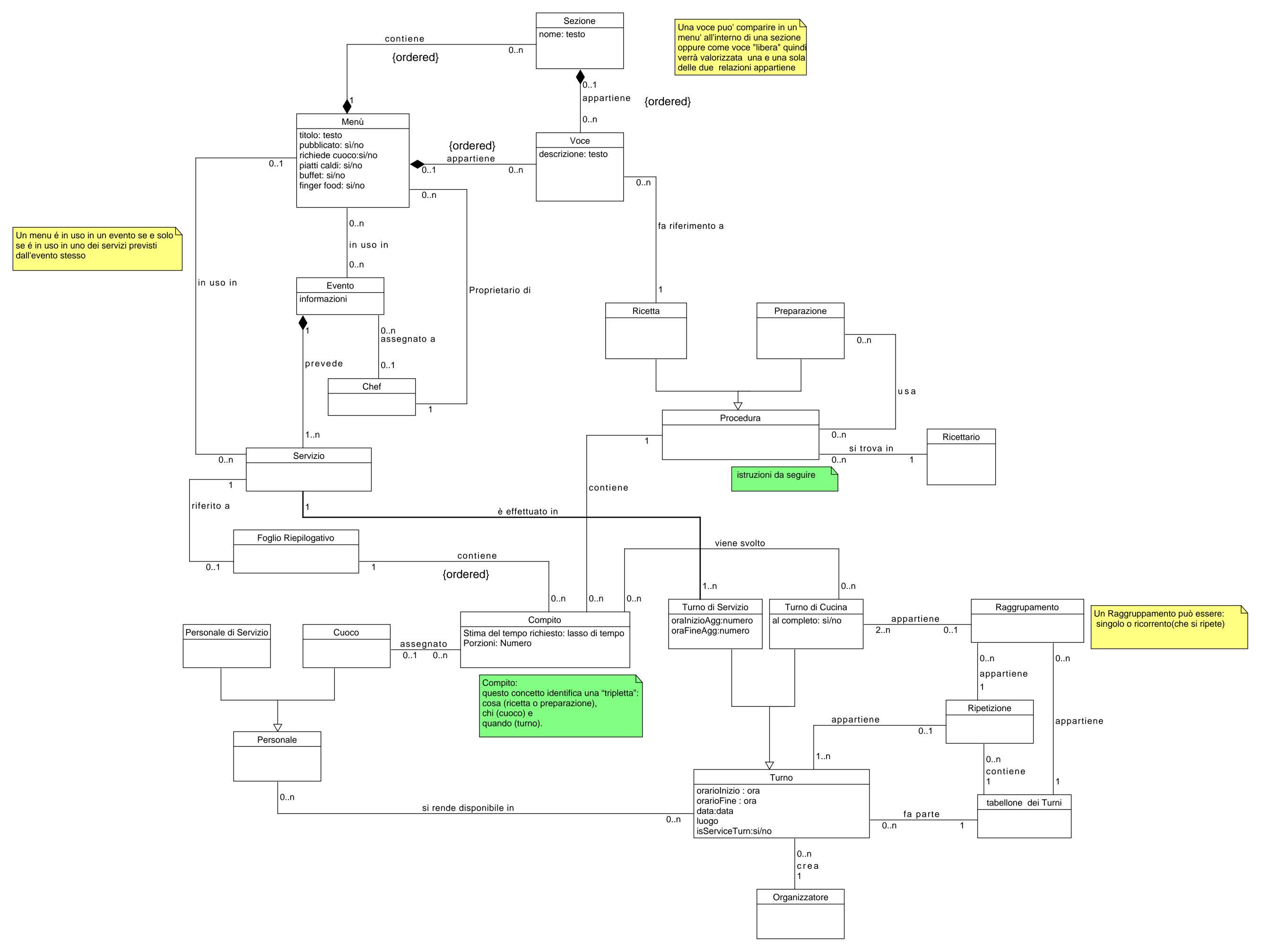
#	Attore	Sistema
5b.1	Cancella compito	Registra modifica
	Termina il caso d'uso	

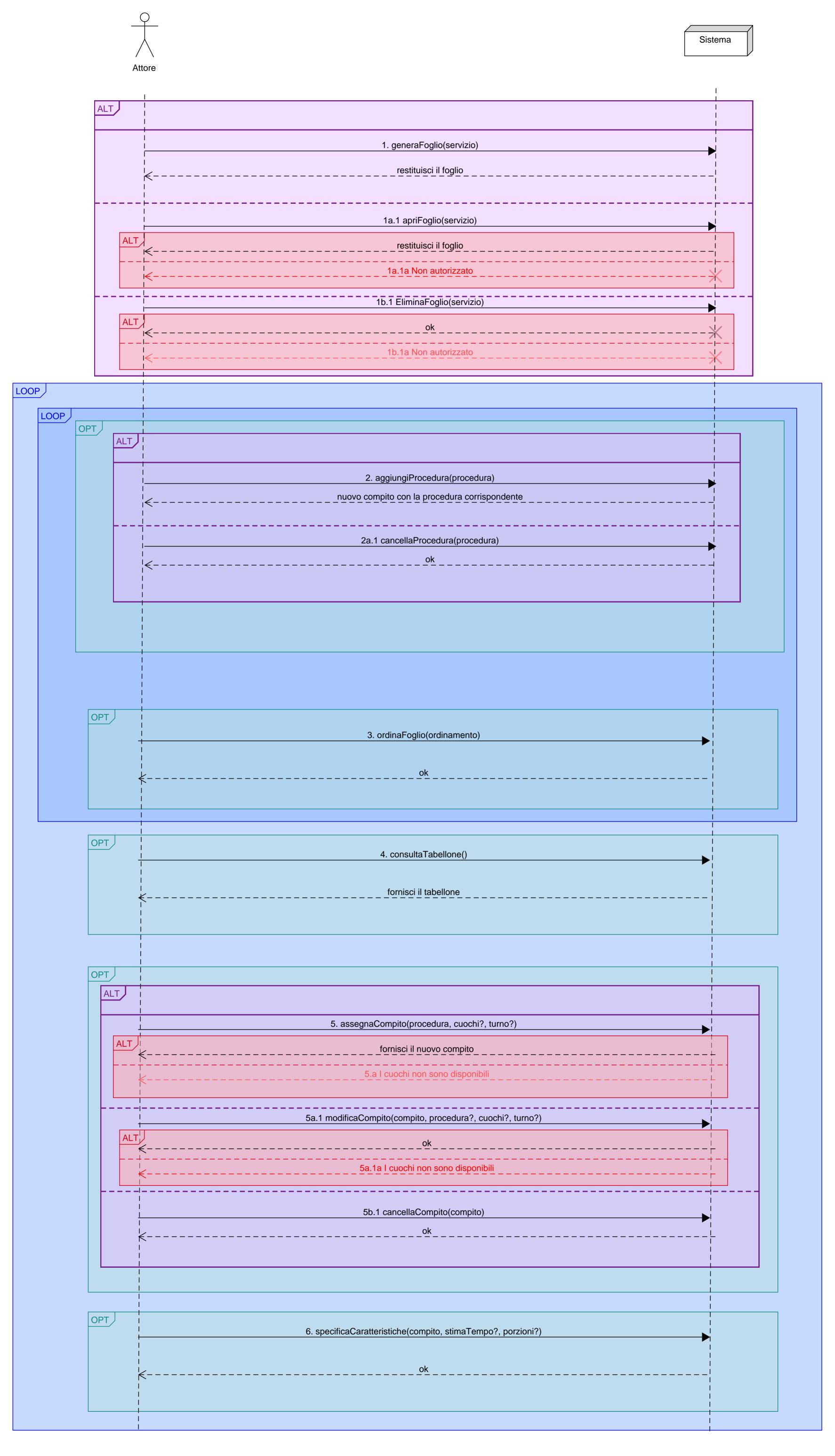
# **Eccezione 5a**

#	Attore	Sistema
5a.1	Assegna compito	Il cuoco che lavora al compito non è disponibile per il turno indicato
	Termina il caso d'uso	

# **Eccezione 5a.1**

#	Attore	Sistema
5a.1a.1	Modifica compito	Il cuoco che lavora al compito non è disponibile per il turno indicato
	Termina il caso d'uso	





### Contratti per lo UC "Gestire I Compiti della cucina"

Pre-condizione generale: L'attore è identificato con un'istanza ch di Chef

#### 1. generaFoglio(evento: Evento, servizio: Servizio)

Pre-Condizioni:

servizio s é previsto in un evento e in cui si usa un menú m di proprietá di ch

Post-Condizioni:

è stata  ${\bf creata}$ una nuova istanza foglio f di Foglio Riepilo gativo

f si riferisce a servizio

Per ogni voce *v* appartenente a *m*:

é stato creato un compito c relativo a v

c é contenuto in f

Per ogni voce v appartenente a una sezione contenuta in m:

é stato creato un compito c relativo a v

c é contenuto in f

#### 1a.1 apriFoglio(servizio: Servizio)

Pre-Condizioni:

Esiste un foglio f che si riferisce a servizio s

Post-Condizioni: - -

#### 1b.1 eliminaFoglio(servizio: Servizio)

Pre-Condizioni:

Esiste un foglio f che si riferisce a servizio

Post-Condizioni:

Per ogni compito *c* di *f*:

c é stato eliminato

#### 2. aggiungiProcedura(procedura: Procedura)

Pre-Condizioni:

Esiste il foglio riepilogativo f

Post-Condizioni:

è stato creata una nuova istanza compito di Compito c

c contiene Procedura

f contiene c

#### 2a.1 cancellaProcedura(procedura: Procedura)

Pre-Condizioni:

Esiste il foglio riepilogativo f

Post-Condizioni:

Per ogni compito c di f:

Se *c* **conteneva** <u>procedura</u>, *c* é stato eliminato

#### 3. ordinaElenco(Ordinamento)

Pre-Condizioni:

Esiste il foglio riepilogativo f

Post-Condizioni:

l'associazione **contiene** fra f e i Compiti é modificata in accordo a <u>ordinamento</u>

#### 4. consultaTabellone()

Pre-Condizioni: —
Post-Condizioni: —

# 5. aggiungiCompito(<u>procedura</u>: Procedura, <u>cuochi</u>?: Elenco di Cuoco, <u>turno</u>?: Turno)

Pre-Condizioni:

Esiste il foglio riepilogativo f

Post-Condizioni:

[Se cuochi è specificato, ma turni no]:

non è cambiato nulla

è stato **creato** un <u>compito</u> c

c contiene la procedura p

per ogni cuoco disponibile in turno:

<u>cuoco</u> è assegnato a c

c si svolge in turno

[se <u>cuochi</u> non è specificato, ma <u>turno</u> si]:

c si svolge in turno

f contiene c

# 5a.1modificaCompito(<u>compito</u>:Compito, <u>procedura</u>?: Procedura, <u>cuochi</u>?: Elenco di Cuoco, <u>turno</u>?: Turno)

Pre-Condizioni:

Esiste il foglio riepilogativo *f* 

Post-Condizioni:

compito contiene la procedura

per ogni <u>cuoco</u> **disponibile in** <u>turno</u>:

<u>cuoco</u> **è assegnato** a <u>compito</u>

<u>compito</u> **viene svolto** in <u>turno</u>

#### **5b.1** cancellaCompito(<u>compito</u>: Compito)

Pre-Condizioni:

Esiste il foglio riepilogativo f contente compito

Post-Condizioni:

il <u>compito</u> è stato cancellato f non contiene più <u>compito</u>

# 6. specificaCaratteristiche(<u>compito</u>: Compito, <u>stimaTempo</u>?: Tempo, <u>porzioni</u>?: Numero)

Pre-Condizioni:

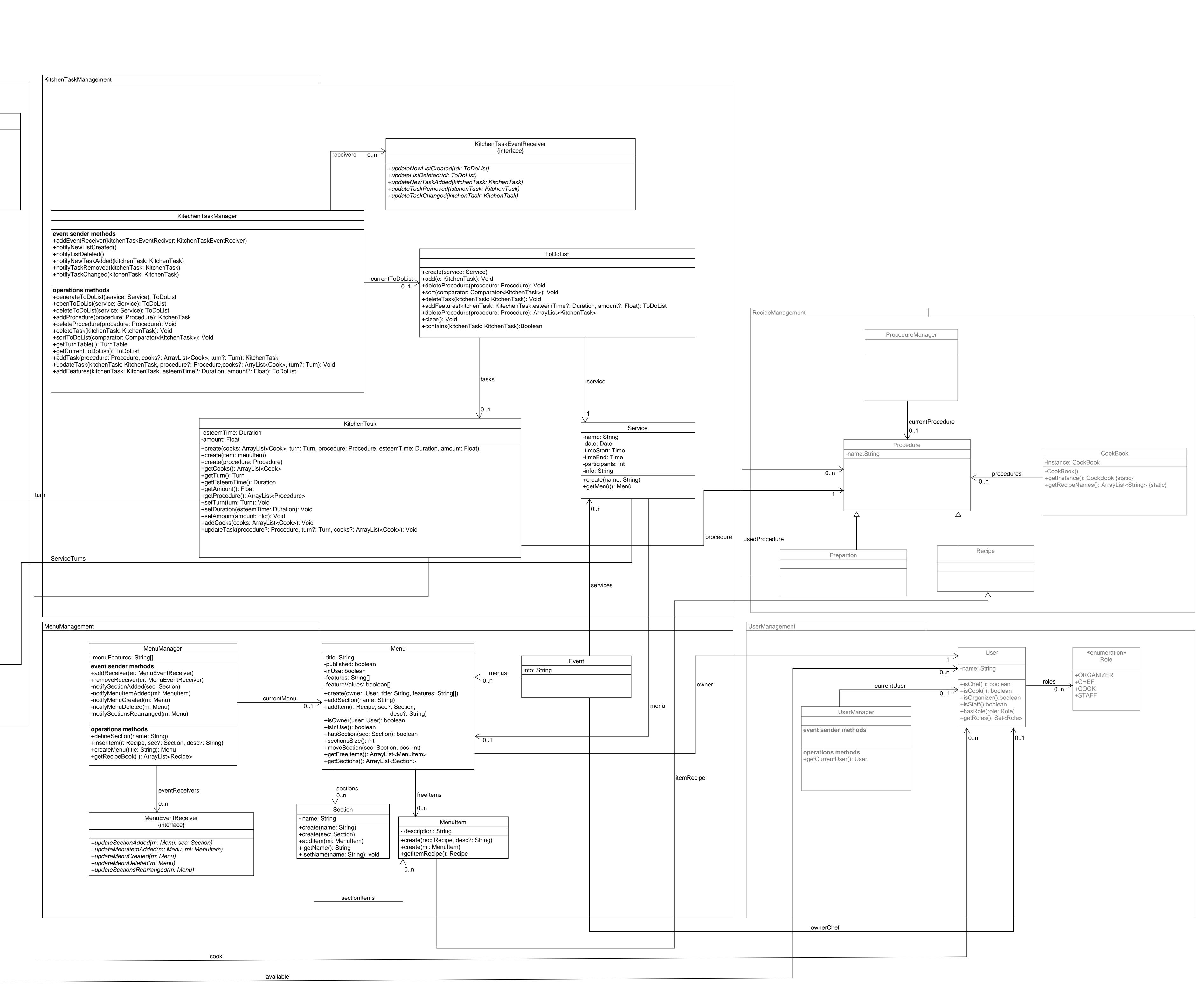
Esiste il foglio riepilogativo <u>f</u>contente <u>compito c</u>

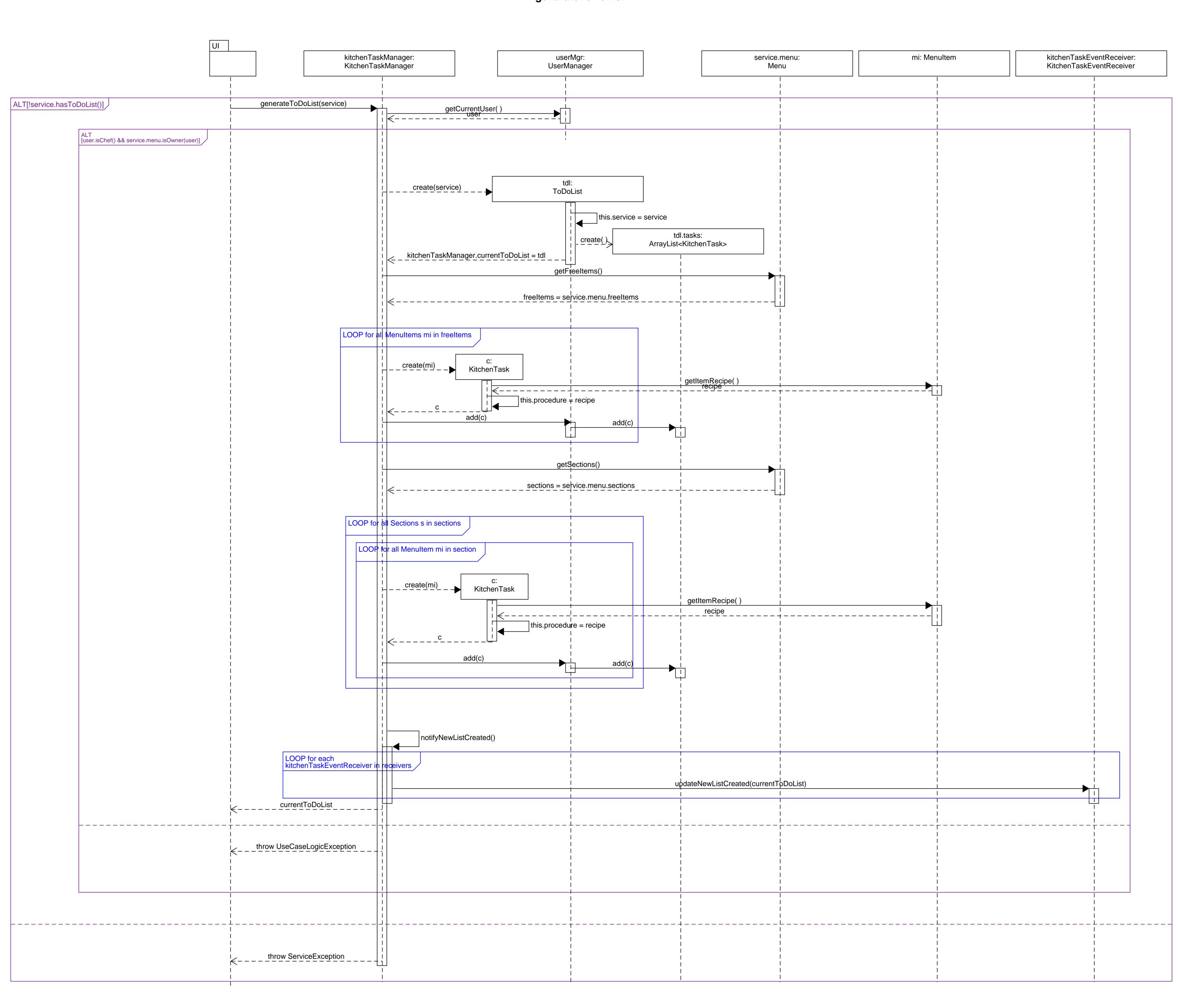
Post-Condizioni:

[se è specificato stimaTempo] <u>c</u>.StimaDelTempoRichiesto = <u>stimaTempo</u> [se è specificata porzioni] <u>c</u>.Porzioni=<u>porzioni</u>

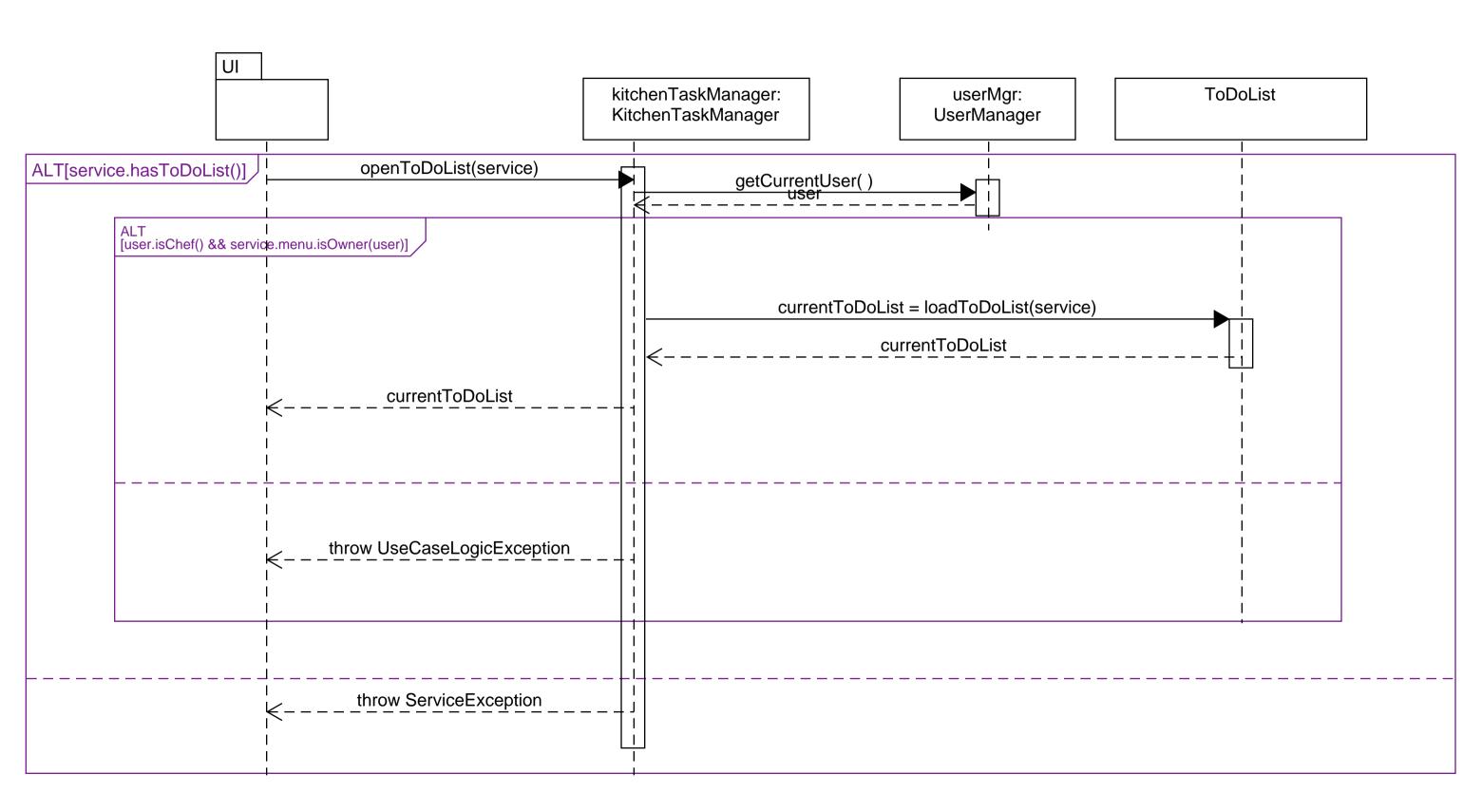
# TurnManagement TurnManager event sender methods +addReceiver(tev: TurnEventReceiver) +notifyModifyTurnRepetiton(turn:Turn) TurnEventReceiver {interface} +removeReceiver(ter: TurnEventReceiver) +notifyModifyTurnRepetiton(turn:Turn) +notifyNewRepetionCreate(repetition: Repeater) +updateModifyTurnRepetiton(turn:Turn) +notifyTurnAdded(turn:Turn) +updateTurnDeleted(turn:Turn) +notifyTurnDeleted(turn:Turn) +updateDeleteGroup(group:Grouper) +updateNewRepetionCreate(turn:Turn, repetion:Repeater) +notifyGroupsCreate(group:Grouper) +notifyDeleteGroup(group:Grouper) +updateDeleteRepetition(turn) +updateNewGroupsCreate(group) operations methods + getTurnTable(): TurnTable + createTurn(date:LocalDate, timeStart:float, imeEnd:float, location?:String,timeStartExtra:Duration, timeEndExtra:LocalDate,isServiceTurn:boolean ):Turn -calcolateCadence(turn.date:LocalDate, cadence:String, endDateRepeat:LocalDate): ArrayList<LocalDate> +createRepetition(turn,cadence:String,endDateRipetition:Date):void +createGroup(turn:Turn,otherTurn:Turn):void +modifyTurn(turn:Turn):turn +deleteTurn(turn:Turn):void +modifyRepetition(turn,NewCadence,NewEndRipetitionDate):void +modifyGroup(group:Grouper,newData:Data):void +deleteGroup(group:Grouper):void +deleteRepetitionTurn(turn:Turn,endDateRipetition:Date?):void +createRipetitionGroup(group,cadence:String,endDateRipetition:Date) +modifyRepetitionGroup(group:Grouper,cadence?,endRipetitionDate?:Date):void +deleteRepetitionGroup(group:Grouper,dateStart:LocalDate,dateEnd:Date):void Repeater -ArrayList<Turn> +create(currentTurn:Turn, listCadence:ArrayList<Date>, endDateRepeat:Date) +modifyRepetition(turn:Turn.NewCadence. turnTable NewEndRipetitionDate:Date) +getTurns():ArrayList<Turn> +create(turn:Turn, cadence, endDateRepeat:Date):void TurnTable +deleteTurn(turn):void repetes - instance: TurnTable -timeStart: float - TurnTable() -timeEnd: float + getInstance(): TurnTable {static} -location: String -isServiceTurn:boolean +addTurn(turn:Turn):void +addRepetition(repetition):void +createTurn(date:Date,timeStart:float,timeEnd:float,location:String, isServiceTurn:boolean):Turn +addGroup(group):void +checkAvailability(turn:Turn):boolean +hasRepetition():boolean +deleteTurn(turn:Turn):void +isServiceTurn():boolean +getAllTurns():ArrayList<Turn> +createRepetition(turn,cadence:String,endDateRipetition:Date):void Grouper | +removeGroup(group):void +deleteTurn(turn:Turn):void -ArrayList<Turn> +modifyTurn(turn:Turn):turn +deleteRepetitioTurn(turn:Turn,endDateRipetition:Date?):void +creatcreateGroup(turn:Turn,otherTurn.Turn):void +modifyRepetition(turn:Turn,NewCadenc:Stinge,NewEndRipetitionDate):void groups +deleteGroup(group):void +getGroup():ArrayList<Turn> turns serviceTurn kitchenTurn -timeStartAdditional: int turnTable -isComplete:boolean -timeEndAdditional: int +createGroup(otherTurn:Turn):void +createTurn(date,timeStart +setGroup(group):void +hasGroup():boolean ,timeEnd:LocalDate,location,AdditionalStartTime:Duration, +createGroup(turn:Turn,otherTurn:Turn):void AdditionalEndTime:Duration):Turn +createRipetitionGroup(group,cadence:String,endDateRipetition:Date) +modifyGroup(group:Grouper,newData:Data):void +deleteGroup(group:Grouper):void +modifyRepetitionGroup(group:Grouper,cadence?,endRipetitionDate?:Date):void +deleteRepetitionGroup(group:Grouper,dateStart:LocalDate,dateEnd:Date):void General module Exception UseCaseLogicException ServiceException

# Design Class Diagram (DCD)

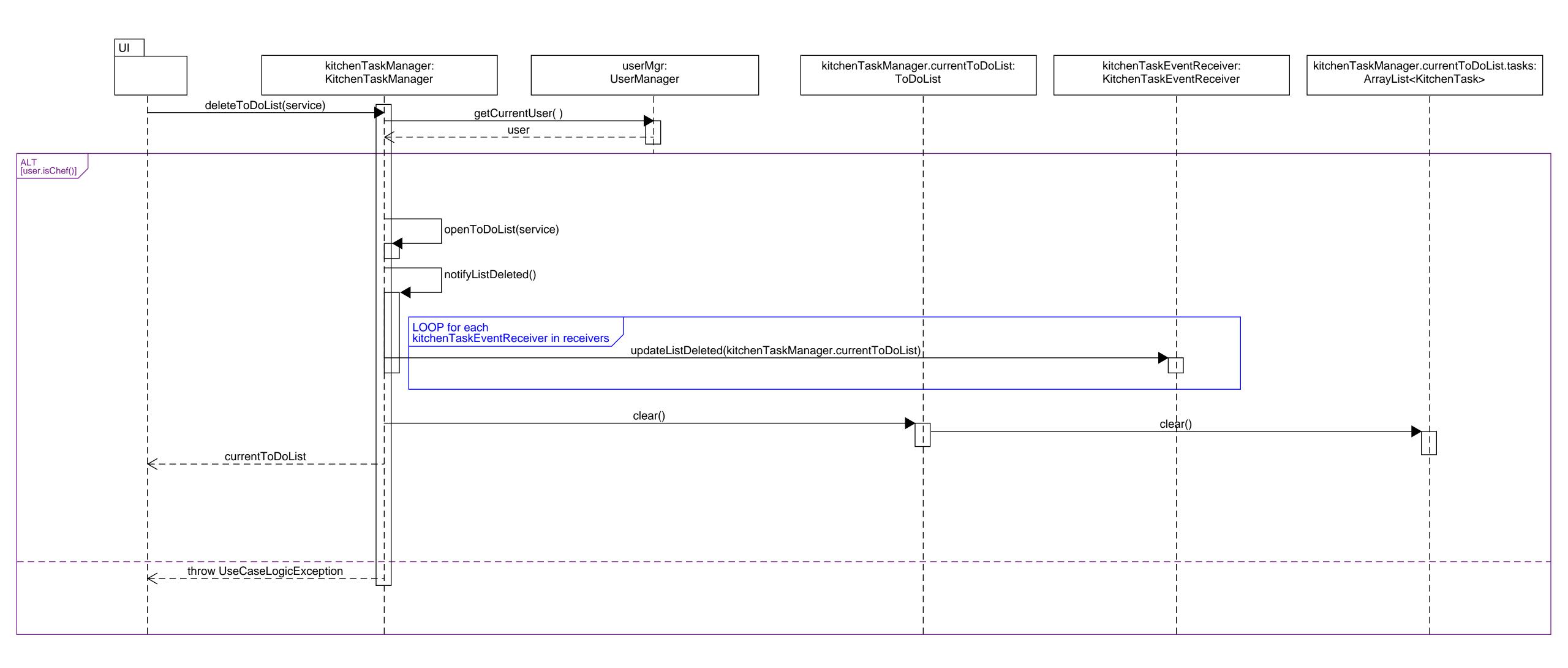


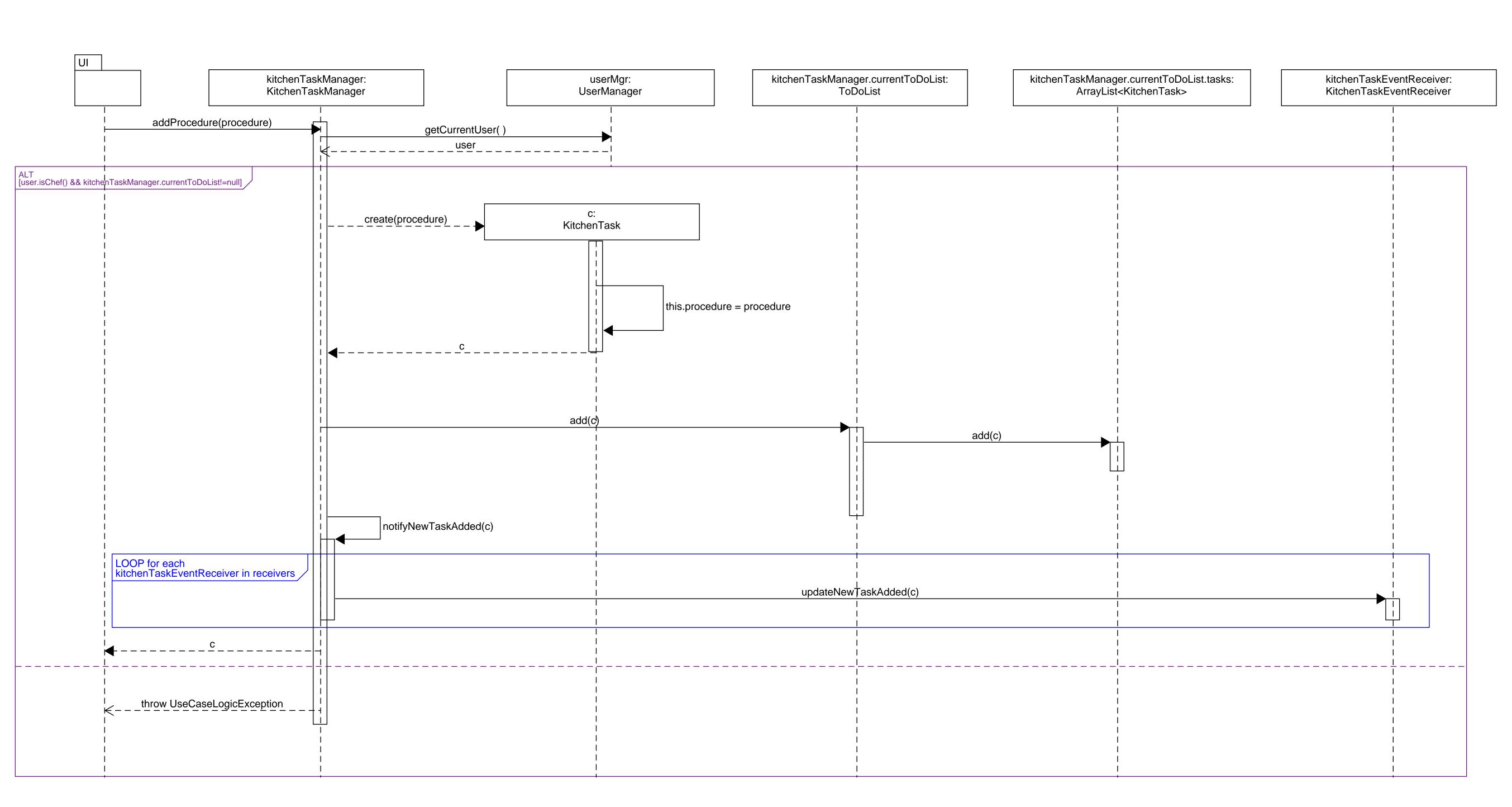


# 1a.1 openToDoList

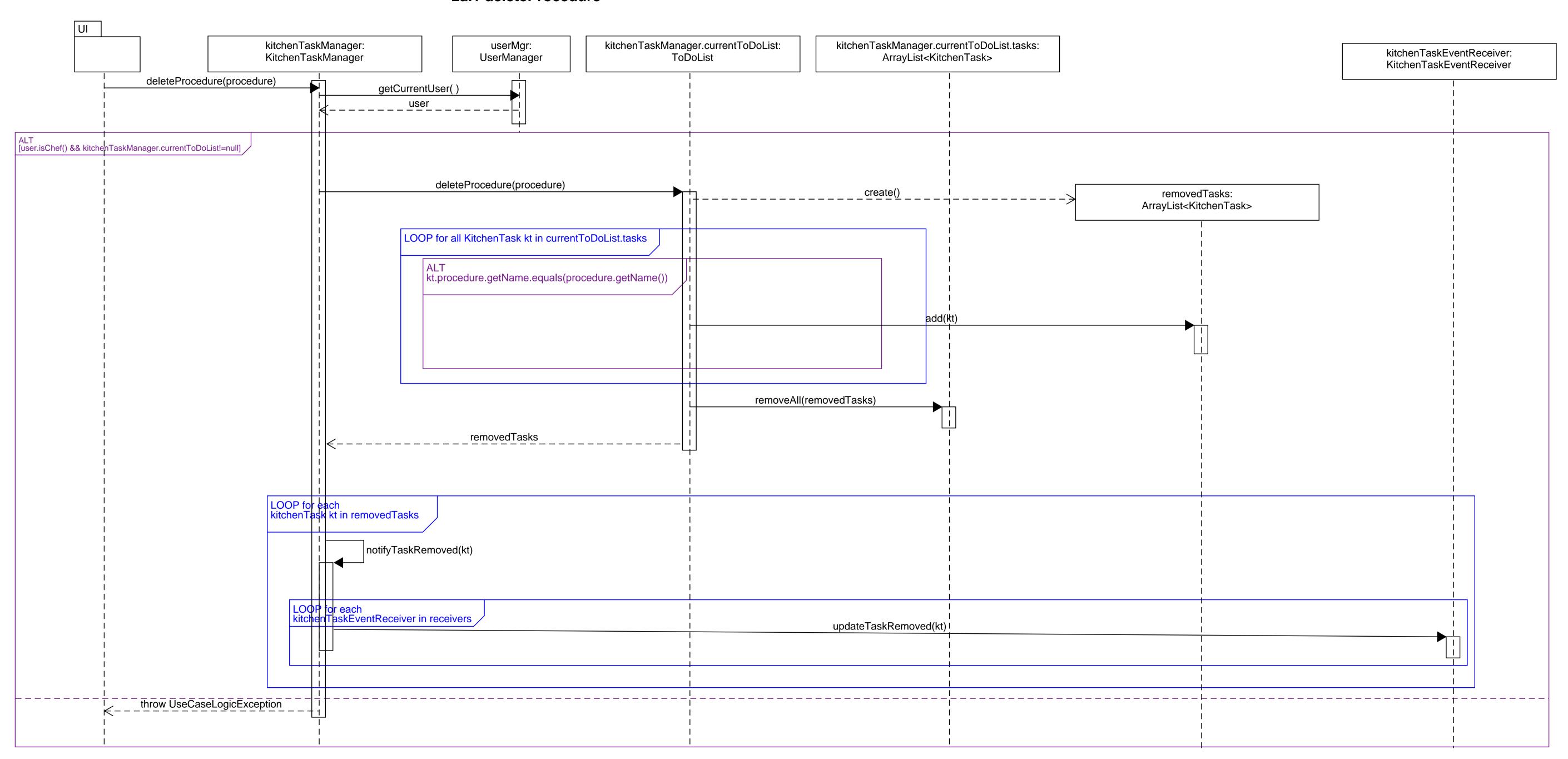


# 1b.1 deleteToDoList

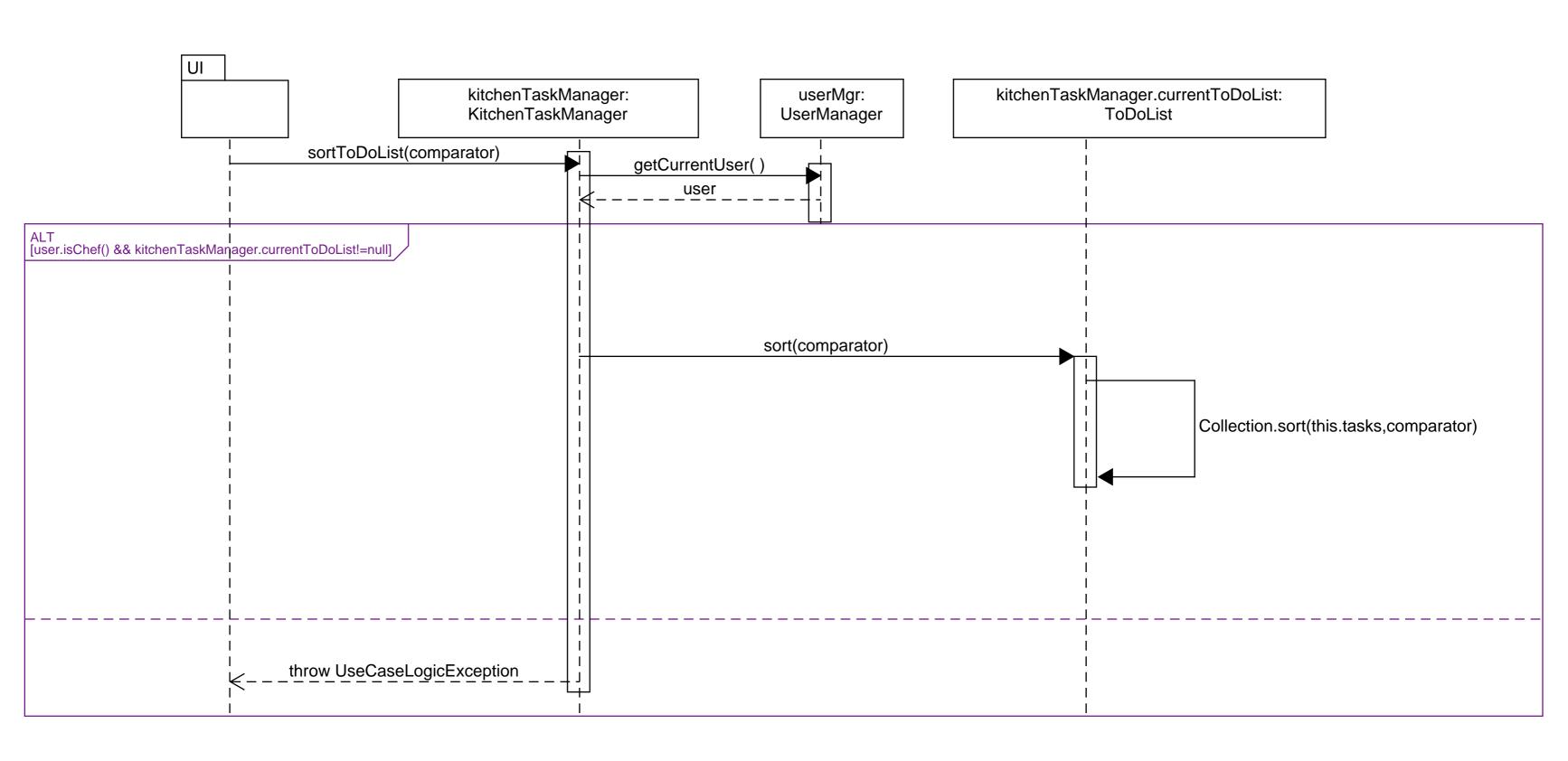




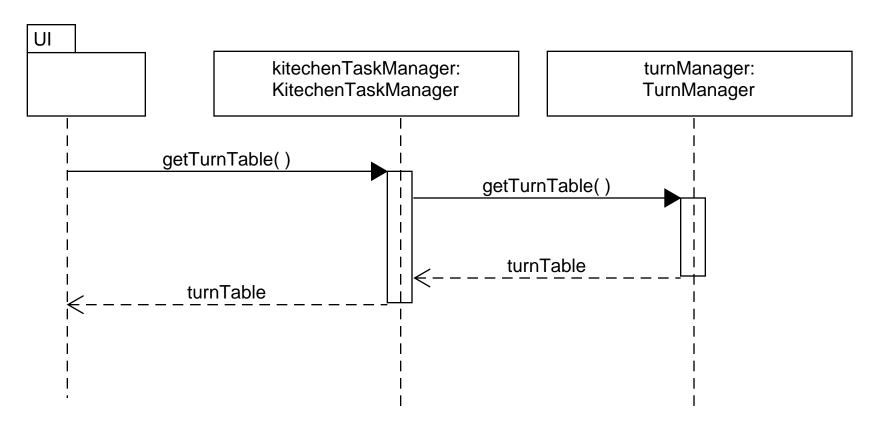
# 2a.1 deleteProcedure

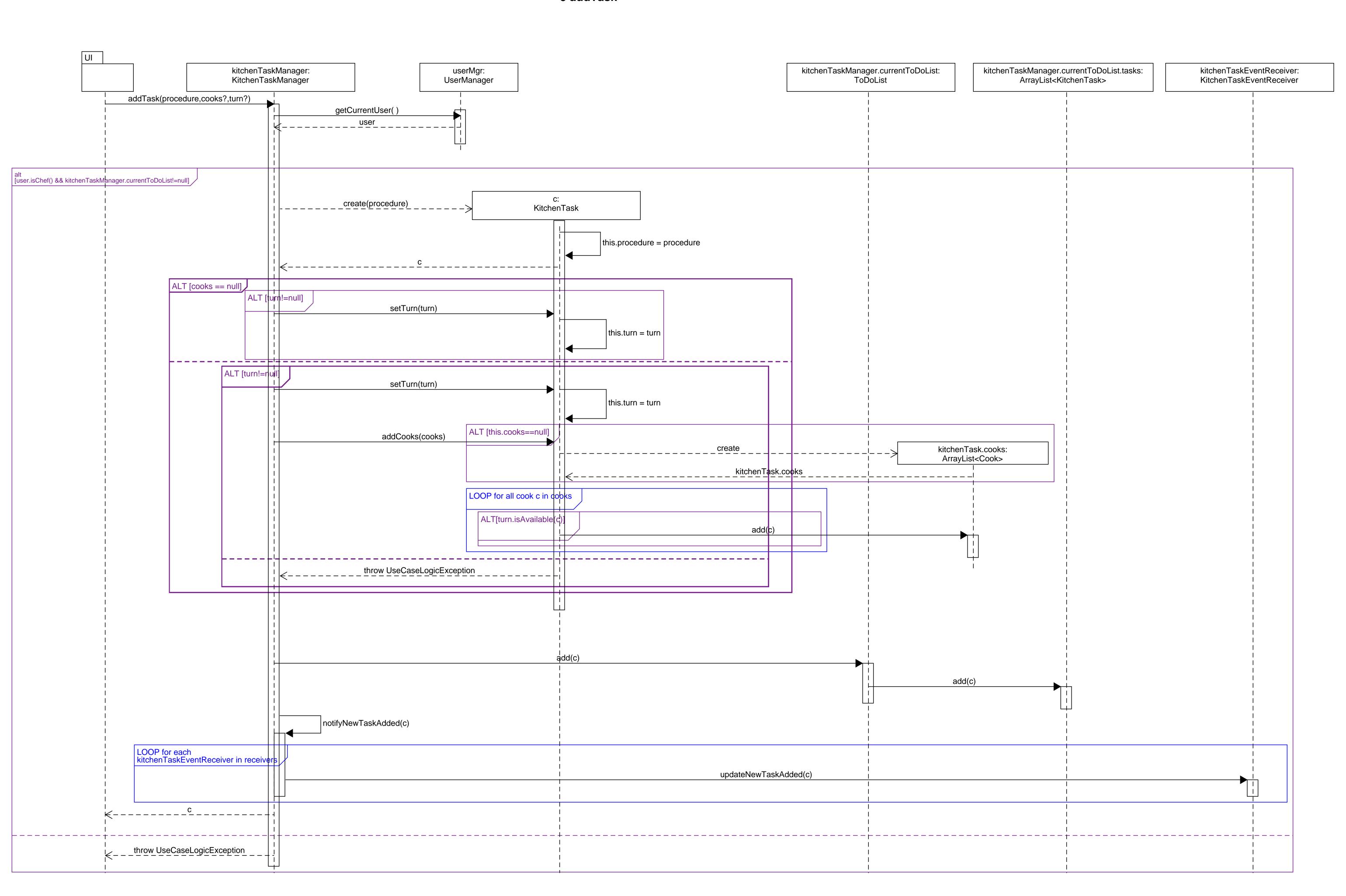


# 3 sortToDoList

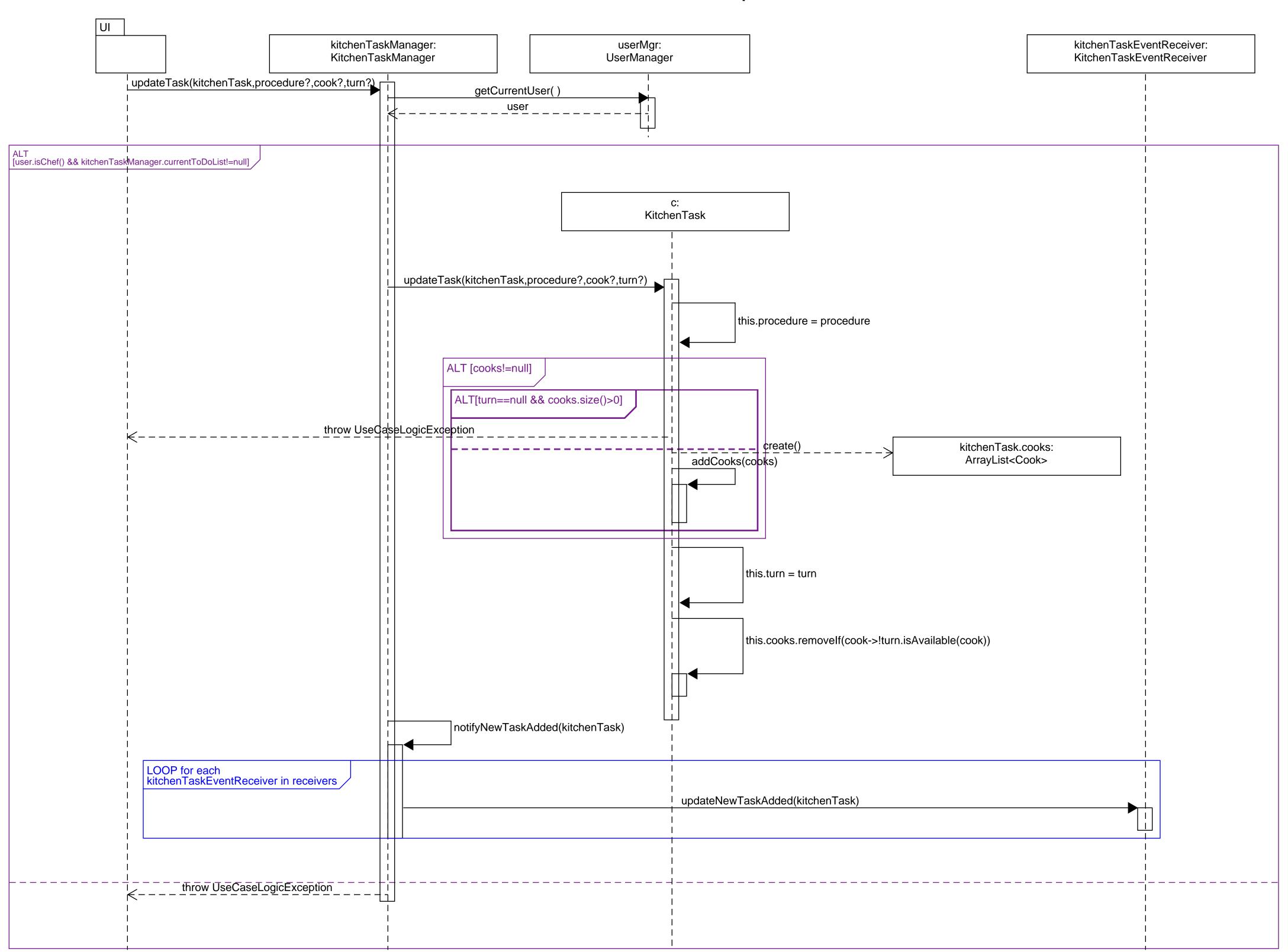


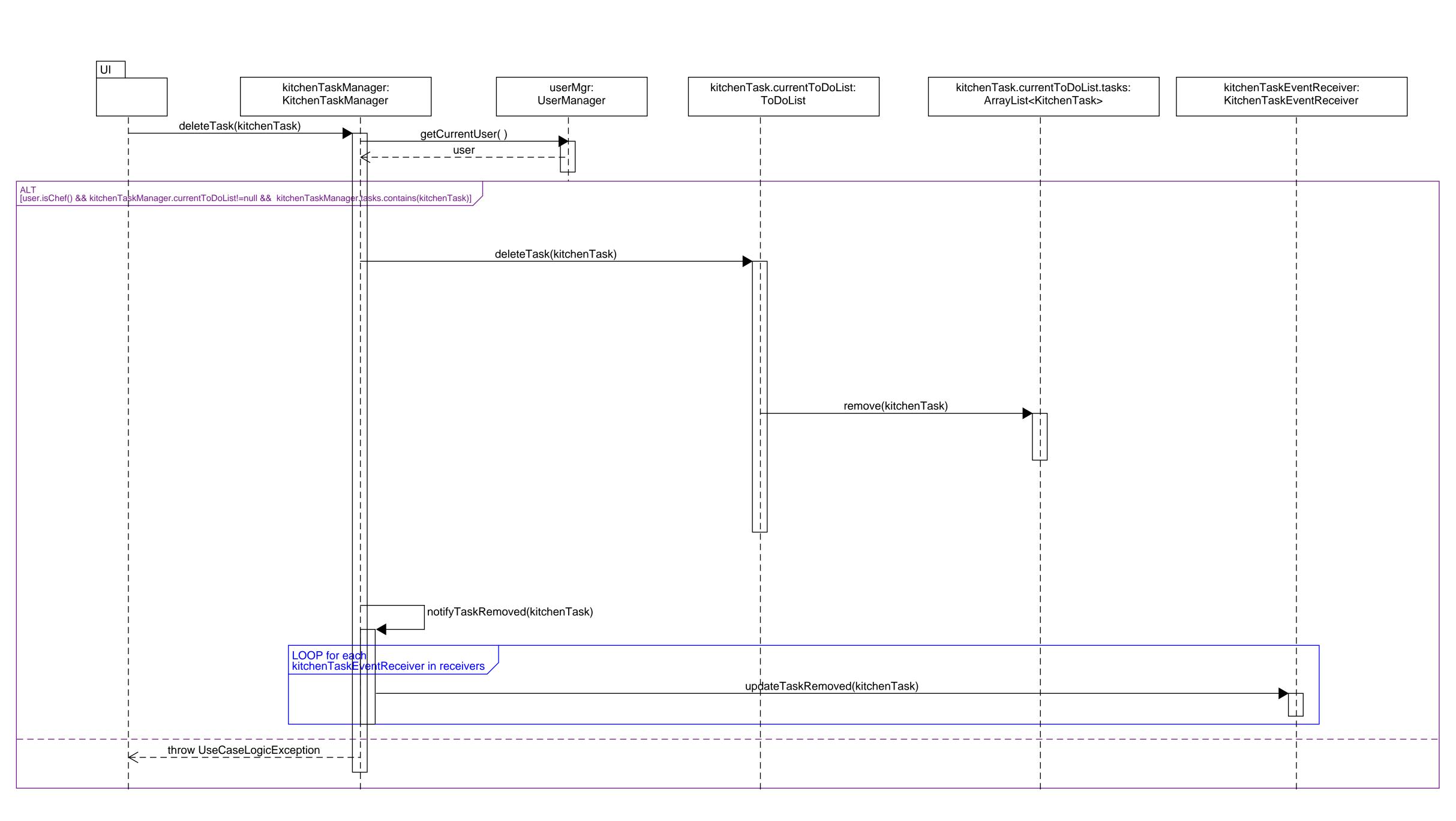
### 4 consultTurnTable





# 5a.1 updateTask





# 6 addFeatures

