Sviluppo Applicazioni Software 20/21

Gianluca Cognigni

748049

Uc Dettagliati

Compiti in cucina

Informazioni generali

Nome caso d'uso: Compiti in cucina
Portata:
Livello:
Attore primario: Chef
Parti Interessate: Cuochi
Pre-condizioni: Aver ricevuto l'incarico di uno o più eventi e che uno o più meno siano stati confermati
Garanzie di successo o post-condizioni: Aver dei compiti assegnati a turni per la preparazione dell'evento, (opzionale) anche dei cuochi assegnati ai vari compiti nei turni

Scenario principale di successo

#	Attore	Sistema
1	Genera il foglio riepilogativo per un servizio di un evento (di cui ha ricevuto l'incarico	Il sistema fornisce un riepilogativo parzialmente precompilato allo chef
	Segue il passo 2 o termina il caso d'uso	
2	**(opzionale)** Aggiunge preparazioni e ricette all'elenco delle cose da fare	Il sistema aggiunge delle preparazioni e ricette alla lista nel foglio riepilogativo
3	**(opzionale)** Ordina l'elenco	Il sistema ordina l'elenco
	Se vuole lavorare su più fogli riepilogativi ripete dal passo 1	
4	***(opzionale)*** *consulta tabellone dei turni*	Il sistema visualizza il tabellone dei turni
5	Assegna un compito specificando cosa (ricetta/preparazione), quando (turno), e (opzionale) chi (cuoco)	Il sistema aggiunge una sezione al riepilogativo con le informazioni date in input e registra anche sul tabellone le info
6	**(opzionale)** Indica una stima del tempo richiesto per lo svolgimento del compito appena assegnato e la quantità/porzioni preparate in un dato assegnamento	Il sistema registra in una sezione del riepilogativo la tempistica per la preparazione ricetta e quantità/porzioni
	Ripete dal passo 4 finchè non è soddisfatto	

Estensione 1a

#	Attore	Sistema
1	Lo chef parte da un foglio riepilogativo esistente (fra quelli dei servizi degli eventi di cui ha ricevuto l'incarico)	Il sistema cerca e fornisce il foglio riepilogativo desiderato da parte dell'utente
	Torna allo scenario principale con il passo 2 o termina il caso d'uso	

Eccezione 1a.1a

#	Attore	Sistema
1	Lo chef parte da un foglio riepilogativo esistente (fra quelli dei servizi degli eventi di cui ha ricevuto l'incarico)	Il sistema non permette la creazione del foglio riepilogativo perchè il menu non è stato approvato
	Uscita dal caso d'uso	

Eccezione 1a.1b

#	Attore	Sistema
1	Lo chef parte da un foglio riepilogativo esistente (fra quelli dei servizi degli eventi di cui ha ricevuto l'incarico)	Il sistema non permette la creazione del foglio riepilogativo perchè l'evento non è stato preso in gestione dallo chef autenticato
	Uscita dal caso d'uso	

Estensione 5a

#	Attore	Sistema
1	Specifica che una ricetta/preparazione è già pronta	Il sistema registra nel compito contenente la ricetta/preparazione che non è da preparare
	Torna allo scenario principale	

Eccezione 5a.1a

#	Attore	Sistema
1	Lo chef parte da un foglio riepilogativo esistente (fra quelli dei servizi degli eventi di cui ha ricevuto l'incarico)	Il sistema non permette la creazione del foglio riepilogativo perchè il menu non è stato approvato
	Uscita dal caso d'uso	

Estensione 5b

#	Attore	Sistema
1	Modifica l'assegnamento	Il sistema modifica l'assegnamento con le modifiche volute
	Torna allo scenario principale	

Estensione 5c

#	Attore	Sistema
1	Elimina l'assegnamento	Il sistema elimina l'assegnamento considerato
	Torna allo scenario principale	

Sezione
nome: testo

Una voce puo' comparire
menu' all'interno di una
oppure come voce "libera"
verrà valorizzata una e
delle due relazioni app

0..n
contiene
1
0..1
appartiene
0..n

Menù
titolo: testo
pubblicato: sì/no
richiede cuoco: sì/no
piatti caldi: sì/no
richiede cucina: sì/no

0..1
appartiene
0..n

Voce
descrizione: tes

in uso in
0..1
0..n
0..n
fa riferimento a
1

Ricetta
nome: testo
autore: testo
descrizione: testo
numero porzioni: n
note: testo
tag: testo

si trova in
0..n
1
0..n
contenuto
0..n

0..n
1..n
Evento
nome: testo
informazioni: te
ricorrente: si/n

Service
name: testo

prevede
1..n
1
1
1
associato a
0..1
0..n
assegnato a
0..1

Preparazione

0..n
0..1
contiene
1..n
si trova in

Ingrediente
nome: testo
quantità: numero

Foglio Riepigolativo
titolo: nome
informazioni: testo

creato da
0..n 0..1

Chef
nome: testo

ha in gestione
associato a
1
0..1
0..1
contiene {Ordered}
0..n
crea
contiene
contiene info uguali
1
0..n
1
0..n
si trova in
1
Ricettario

Mansione

Se l'attributo "completato" di un dato compito
è impostato a si, allora la mansione è collegata
per forza ad un turno ed un cuoco

Organizzatore
nome: testo

1
1
0..1
1
1
consiste in
0..1
0..n
si trova in
1
ElencoIngredienti

Tabellone Turni
/scadenzaGlobale: Data
currentDate: testo

Compito
quantità: numero
completato: si/n
stima: numero

0..n
Cuoco
nome: testo

0..n
eseguito
0..1

contiene
0..n
eseguita in
0..n
0..1
lavora in
0..n
0..n
gestisce

Turno Preparatorio
/parteDiBlocco: si/no

Turno
/data: Data
fasciaOraria: numero
/scadenza: Data
posizioneLavori: testo

0..n
coordina
da disponibilita
0..n 0..n

Personale di Servizio

0..n
raggruppati in
0..1

Blocco

coordina
1
1
0..n
0..n
coordina

Personale di Sala
nome: testo

Calendario
dataCorrente: Data

possiede

Turno di servizio
tempoAggiuntivoPrima: numero
tempoAggiuntivoDopo: numero

1
coordina
0..1
0..1

Ricorrenza
/dataInizio: Data
/dataFine: Data
cadenza: {settimanale, mensile, annuale}

SSD Cucina

Diagramma di Sequenza di Sistema per UC
"Compiti in cucina"
Scenario principale di successo

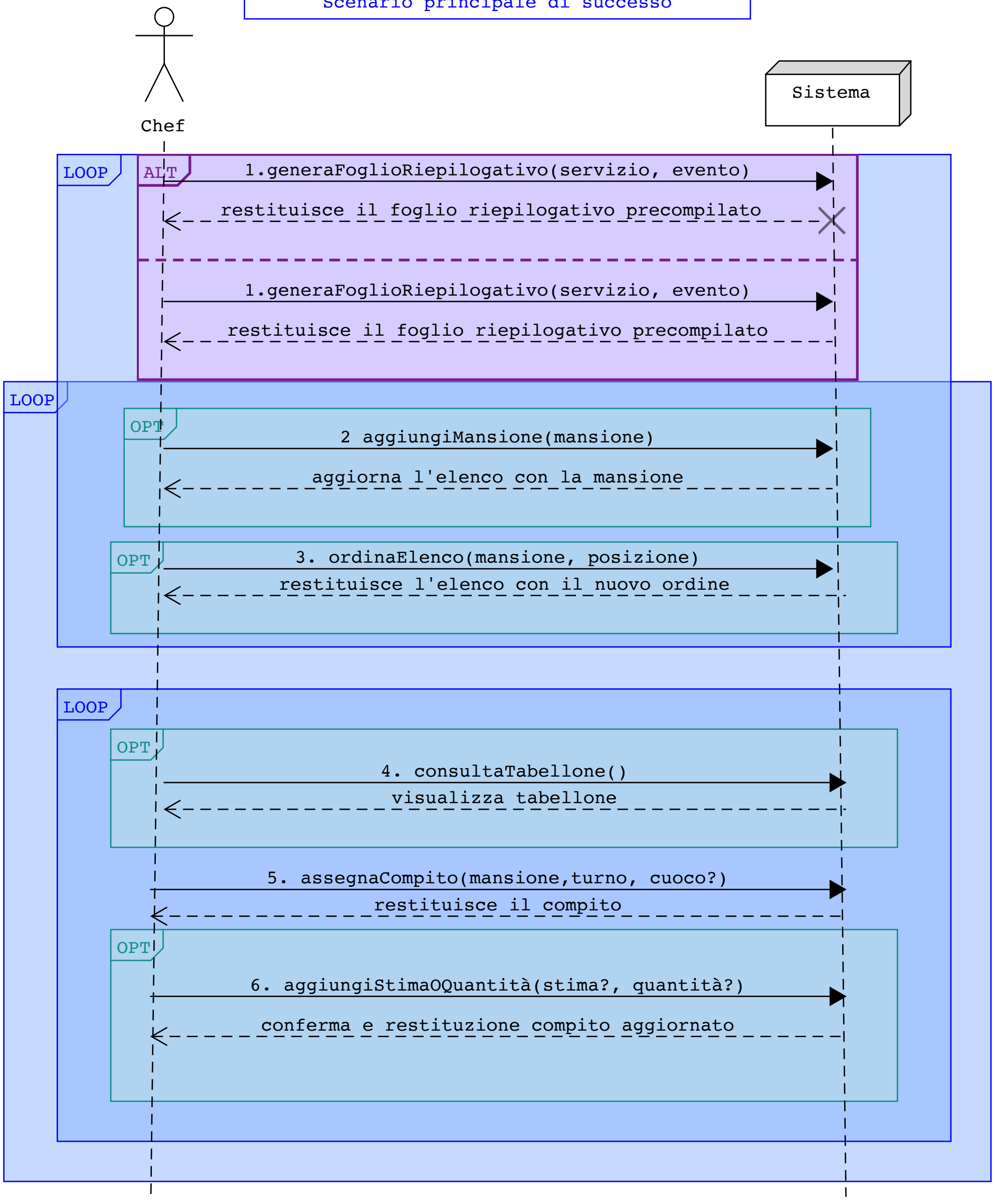


Diagramma di Sequenza di Sistema per UC "Compiti in cucina" Estensione 1a

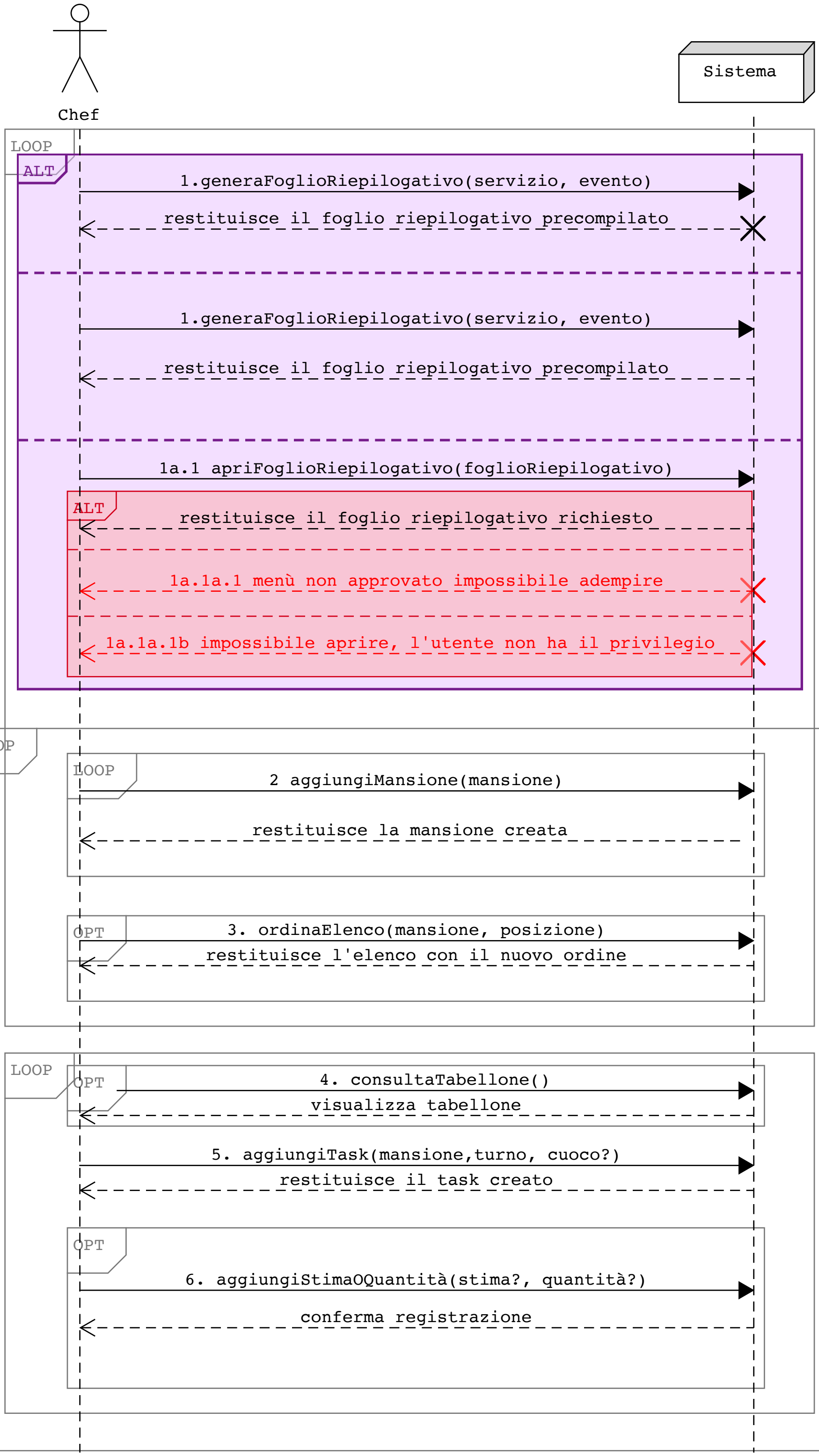
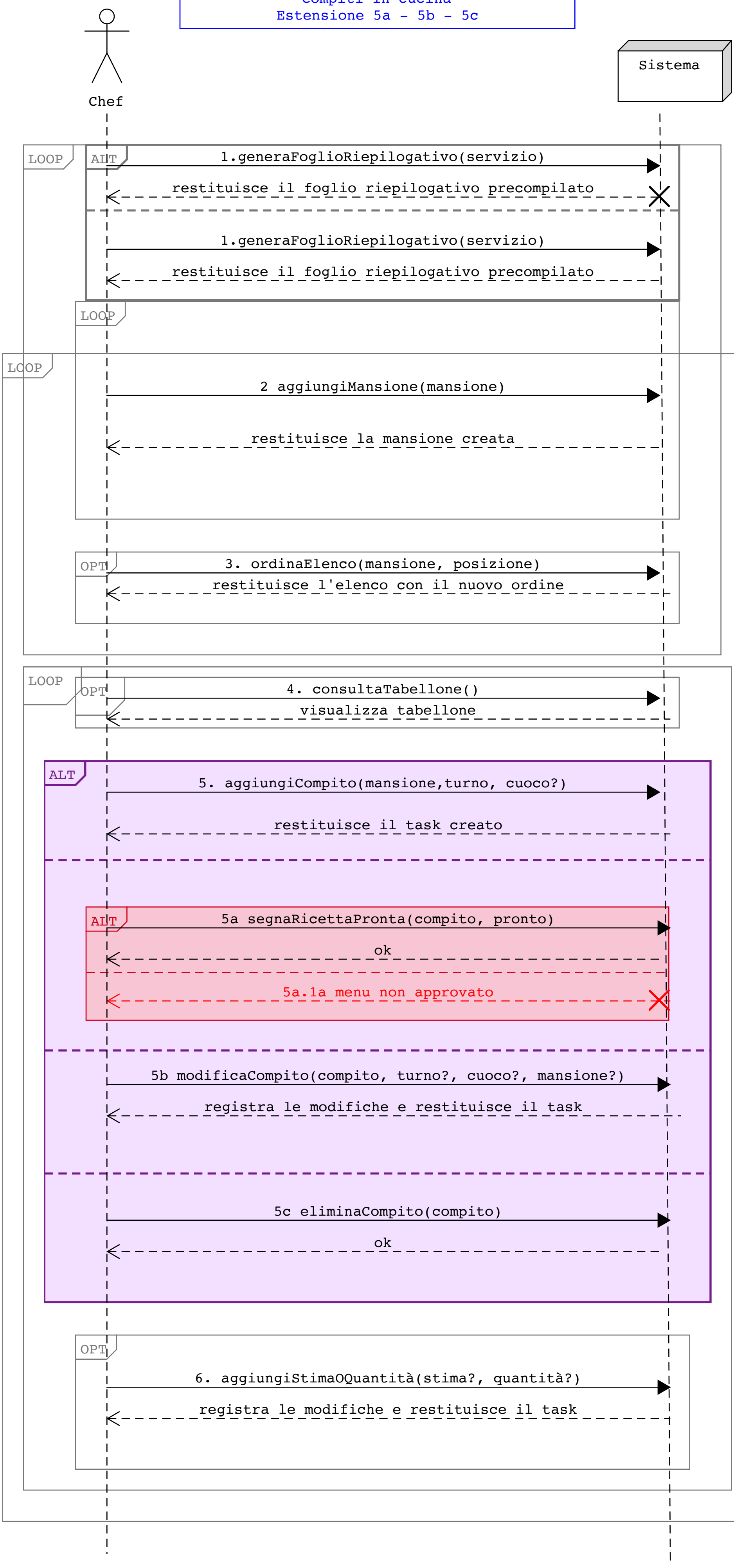


Diagramma di Sequenza di Sistema per UC "Compiti in cucina" Estensione 5a - 5b - 5c



Contratti Cucina

Scenario Principale

Precondizioni Generali Compiti in Cucina

- l'attore è identificato con un'istanza *ch* di Chef

1. GeneraFoglioRiepilogativo(evento: Evento, servizio: Servizio)

pre-condizione:

- che *evento* **preveda** *servizio*
- che *evento* **sia assegnato a** *ch*

post-condizione:

- genera un istanza *fo* di Foglio riepilogativo precompilato
- *fo* **è creato da** *ch*
- *fo*.incarico = *ev*.nome
- *fo*.informazioni = *ev.*informazioni
- l'istanza *fo* è **associata** a *serv*

2. aggiungiMansione(mansione: Mansione)

pre-condizione:

- è in corso la definizione di un Foglio Riepilogativo *fo*

post-condizione:

- *fo* **contiene** *mansione*

3. ordinaElenco(mansione: Mansione, posizione: numero)

pre-condizione:

- è in corso la definizione di un Foglio Riepilogativo *fo*

post-condizione:

- *fo* ***contiene {posizione}*** *mansione*

4. consultaTabellone()

pre-condizione:

- Esistenza di un'istanza *fo* di Foglio Riepilogativo
- Esistenza di un'istanza *tab* di Tabellone Turni
- l'istanza *fo* di Foglio Riepilogativo **contiene info uguali** a *tab*
- *tab* ***è associato*** dall'istanza *serv* di Service

post-condizione:

- Nessuna, è un'operazione di interrogazione

5. aggiungiCompito(mansione: Mansione, turno: Turno Preparatorio, cuoco?: Cuoco)

pre-condizione:

- è in corso la definizione di *fo*
- [in caso cuoco sia specificato] *cuoco* **lavora in** *turno*

post-condizione:

- crea un'istanza *com* di Compito
- *com* **consiste in** *mansione*
- *com* **eseguita in** *turno*
- [in caso cuoco sia specificato] *com* **eseguito** da *cuoco*

6. aggiungiStimaOQuantità (stima?: numero, quantità? numero)

pre-condizione :

- sia in corso la definizione dell'istanza *com* di Compito

post-condizione:

- [in caso stima sia specificato] *com.*stima = stima
- [in caso stima sia specificato] *com.*quantità = quantità

Estensione 1a

Precondizioni Generali Compiti in Cucina

- l'attore è identificato con un'istanza *ch* di Chef

1a apriFoglioRiepilogativo(foglioRiepilogativo: Foglio Riepilogativo)

pre-condizione:

- *foglioRiepilogativo* **creato da** *ch* istanza di Chef

post-condizione:

—-

Estensione 5abc

Precondizioni Generali Compiti in Cucina

- l'attore è identificato con un'istanza *ch* di Chef

5a segnaCompitoPronto(compito: Compito, pronto: si)

pre-condizione:

- è in corso la definizione di un Foglio Riepilogativo *fo*

post-condizione:

- compito.completato = si

5b modificaCompito(compito: Compito, turno?: Turno Preparatorio, cuoco?: Cuoco, mansione?: Mansione)

pre-condizione:

- è in corso la definizione di *fo* istanza di Foglio Riepilogativo
- [in caso cuoco sia specificato] *cuoco* **lavora in** *turno*

post-condizione:

- (se turno specificato)*compito* **eseguito in** *turno*
- (se *cuoco* specificato) *compito* **eseguito da** *cuoco*
- (se *mansione* specificato) *compito* **consiste in** *mansione*

5c eliminaCompito(compito: Compito)

pre-condizione:

- è in corso la definizione di *fo* istanza di Foglio Riepilogativo

post-condizione:

- *fo* ***non contiene*** *compito*
- *compito* viene eliminato

Shift Management

ShiftManagementManager

event sender methods
+addReceiver(er: ShiftBoardReceiver)
+removeReceiver(er: ShiftBoardReceiver)

+notifyCreatedKitchenShift(shift: Shift)
+notifyCreatedServiceShift(shift: Shift)
+notifyModifiedShift(shift: Shift)
+notifyGlobalDeadlineSet(shiftBoard: ShiftBoard, date:Date)
+notifyLocalDeadlineSet(shift: Shift)
+notifyShiftDelete(shift: Shift)
+notifyAvailabilityGiven(shift: Shift, usr: User)
+notifyRecurrenceCreated(rec:Recurrence, shiftsOnDisplay: ArrayList<Shift>)
+notifyAvailabilityGiven (usr: User,shift: Shift)
+notifyRecurrentAvailabilityGiven(usr: User, shifts: ArrayList<Shifts>)
+notifyAvailabilityTaken(usr: User, shift: Shift)
+notifyRecurrentAvailabilityTaken(usr: User, shifts: ArrayList<Shifts>

operations methods
+checkEvent(ev: Event, serv: Service)
+getBoard(shiftBoard: ShiftBoard)
+setCurrentBoard(shiftBoard: ShiftBoard)
+setCurrentShift(shift: Shift)
+createBlock(shifts: ArrayList<KitchenShift>)
+modifyBlock(block: Block, shift: Shift)
+modifyShift(shift: Shift, beginTime?: String, endTime?: String, beforeHandTime?: Integer, aftermathTime?: Integer, location?: String)
+setLocalDeadline(shift: Shift, date: Date)
+createRecurrentBlocks(shift: Block, rec: Recurrence)
+createRecurrentBlocks(shift: Shift, rec: Recurrence)
+addShift(shift: Shift)
+createKitchenShift(date: Date, beginTime: Integer, endTime:Integer, location?: String)
+createServiceShift(date: Date, beginTime: Integer, endTime:Integer, afterMathTime?: Integer, beforehandTime?: Integer, location?: String)
+deleteShift(shift: Shift)
+checkShiftOverlap()
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

currentServiceBoard 0..1

ShiftBoard
-event: Event
-service: Service
-globalEndingDate: Date
-currentDate: Date
+createKitchenShift(date: Date, beginTime: Integer, endTime:Integer, location?: String)
+createServiceShift(date: Date, beginTime: Integer, endTime:Integer, afterMathTime?: Integer, beforehandTime?: Integer, location?: String)
+createBlock(shifts: ArrayList<KitchenShift>)
+setGlobalDeadline(date: Date)
+giveAvailability(shift: Shift, usr: User)
+checkShiftOverlap()
+tableSize()
+deleteShift(shift:Shift)
+getShift(shift: Shift)
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

shiftsOnDisplay 0..n

Shift
«Interface»
+setLocation;
+setAvailability;
+setLocalDeadline;
+isAvailable;
+getAvailabilitySize;
+isDeadlineSurpassed;
+createRecurrence;
+isRecurrent;
+deleteAvailability;

0..n currentShift 0..1
recurringShifts 0..n
recurrence 0..1

Recurrence
-startDate: Date
-endDate: Date
-frequence: {weekly, monthly, yearly}
+addToDisplayedShifts(shift: Shift)
+createShift(date: Date)
+deleteRecurrentAvailability(usr: User)
+giveRecurrentAvailability(usr: User)

ServiceShift
id: Integer
location; String
date: Date
beginTime: String
endTime: String
deadline: Date
aftermathTime: Integer
beforehandTime: Integer
availableStaffList: ArrayList<User>
recurrenceId: Integer
+setLocalDeadline(date: Date)
+setAvailability(usr: User)
+getAvailabilitySize()
+createRecurrence(rec: Recurrence)
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

KitchenShift
id: Integer
location; String
date: Date

beginTime: String
endTime: String

deadline: Date

availableStaffList: ArrayList<User>
recurrenceId: Integer
+setLocalDeadline(date: Date)
+setAvailability(usr: User)
+setRecurrenceId()
+getAvailabilitySize()
+isRecurrent(rec)
+createRecurrence(rec: Recurrence)
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

Block
id: Integer

date: Date

begintime: Integer
endTime: Integer

deadline: Date
availableStaffList: ArrayList<User>
+addShift(shift: KitchenShift)
+setLocalDeadline(date: Date)
+setAvailability(usr: User)
+getAvailabilitySize()
+createRecurrence(rec: Recurrence)
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

eventReceivers 0..n

ShiftReceiver
{interface}
+*updateCreatedKitchenShift(shift:Shift)*
+*updateCreatedServiceShift(shift:Shift)*
+*updateModifiedShift(shift: Shift)*
+*updateLocalDeadlineSet(shift: Shift)*
+*updateShiftDeleted(shift: Shift)*
+*updateLocalDeadline(shift: Shift)*
+*updateModifiedBlock(block: Block)*
+*updateAvailabilityGiven(shift: Shift, usr: User)*
+*updateRecurrenceCreated(rec:Recurrence, shiftsOnDisplay: ArrayList<Shift>)*
+*updateRecurrentAvailabilityGiven(usr: User, shifts: ArrayList<Shifts>)*
+*updateAvailabilityGiven (usr: User,shift: Shift)*
+*notifyAvailabilityTaken(usr: User, shift: Shift)*
+*notifyRecurrentAvailabilityTaken(usr: User, shifts: ArrayList<Shifts>*

current Board

availabilities

shiftInUse 0..1

RecipeManagement

RecipeManagement

Event
Service
0..n
1 currentEvent
1 currentService
Event Manager

itemRecipe

Recipe
-name: String
0..n recipes
RecipeManager
+getRecipeBook(): ArrayList<Recipe>
+getPreparationBook(): ArrayList<Preparation>
Kitche Job
Preparation
- name: String
contains 0..n

jobList 0..n 1
job

Cucina Management

Summary Sheet
+addTask(kitchenJob: KitchenJob, coo?: User, sft: Shift)
+modifyTask(tsk: Task, kiS?: Shift, coo?: User)
+orderJobList(kitchenJob: KitchenJob, pos: int, originalPos: int)
+deleteTask(tsk: Task)
+markTaskReady(tsk: Task)
+addEstimateOrQuantity(tsk: Task, est? Integer, qt?: Integer)
+getKitchenJobPosition(job: KitchenJob)
+addKitchenJob(kitchenJob: KitchenJob)

itemTask 0..n

Task
id: Integer
quantity: Integer
estimate: Integer
completed: boolean
+markReady()
+addEstimateOrQuantity(est?: Integer, qt?: Integer)
+modifyTask(kitchenJob?:KitchenJob, shft?: Shift, coo?: User)

0..1 currentSheet

Summary Sheet Manager

event sender methods
+addReceiver(er: SummarySheetReceiver)
+removeReceiver(er: SummarySheetoReceiver)
-motifyTaskReady(sumSheet: SummarySheet, tsk: Task)
-notifyTaskCreated(sumSh: SummarySheet, tsk: Task)
-notifyTaskModified(sumSheet: SummarySheet, tsk: Task)
-notifyKitchenJobAdded(sumSh: SummarySheet,kitchenJob: KitchenJob)
-notifyTaskDeleted(sumSheet: SummarySheet, tsk: Task)
-notifyListRearranged(obj: Objective, pos: Integer)
-notifyEstimateOrQuantity(tsk: Task, est?: Integer, qt?: Integer)
-notifySummarySheetGenerated(sumSh: SummarySheet)

operations methods
+generateSummarySheet(ev: Event, serv: Service)
+openSummarySheet(sumSheet: SummarySheet)
+openShiftBoard(shiftBoard: ShiftBoard)
+getShiftBoard(kiSBoard: ShiftBoard)
+addTask(kitchenJob: Kitchen Job, kiS: Shift, coo? User)
+addKitchenJob(kitchenJob: KitchenJob)
+orderJobList(job: KitchenJob, posWanted: int, originalPos: int)
+markTaskReady(tsk: Task)
+addEstimateOrQuantityToTask(est? Integer, qt?: Integer)
+modifyTask(tsk?: Task, kitchenJob?: KitchenJob, kiS?: Shift, coo?: User)
+deleteTask(tsk: Task)
+getBoard(shiftBoard: ShiftBoard)
+setCurrentBoard(shiftBoard: ShiftBoard)
+setCurrentTask(tsk: Task)
+setCurrebtSheet(sumSh: SummarySheet)
+markTaskReady(tsk: Task)

SummarySheetReceiver
{interface}
+*updateSumSheetGenerated(sumSh: Summary Sheet)*
+*updateTaskCreated(tsk: Task, sumSheet: SummarySheet)*
+*updateModifiedTask(sumSheet: SummarySheet, tsk: Task)*
/+updateKitchenJobAdded(sumSh: SummarySheet, kitchenJob: KitchenJob)
+*updateTaskDeleted(sumSh: Summary Sheet, tsk: Task)*
+*updateListsRearrang(sumSheet: Summary Sheet, obj: Objective, pos: int,)*
/+updateTaskReady(sumShet: Summary Sheet, tsk: Task)/
+*updateEstimateOrQuantity(tsk: Task, est?: Integer, qt?: Integer)*
+*SummaryGenerated(sumSh: SummarySheet)*

0..n eventReceivers

cook

currentUser

owner

Design Class Diagram (DCD)

MenuManagement

MenuManager
-menuFeatures: String[]
event sender methods
+addReceiver(er: MenuEventReceiver)
+removeReceiver(er: MenuEventReceiver)
-notifySectionAdded(sec: Section)
-notifyMenuItemAdded(mi: MenuItem)
-notifyMenuCreated(m: Menu)
-notifyMenuDeleted(m: Menu)
-notifySectionsRearranged(m: Menu)
operations methods
+defineSection(name: String)
+inserItem(r: Recipe, sec?: Section, desc
+createMenu(title: String): Menu
+getRecipeBook( ): ArrayList<Recipe>

currentMenu 0..1

Menu
-title: String
-published: boolean
-inUse: boolean
-features: String[]
-featureValues: boolean[]
+create(owner: User, title: String, fea
+addSection(name: String)
+addItem(r: Recipe, sec?: Section,
+isOwner(user: User): boolean
+isInUse(): boolean
+hasSection(sec: Section): boolean
+sectionsSize(): int
+moveSection(sec: Section, pos: int)

sections 0..n

Section
name: String
+create(name: String)
+create(sec: Section)
+addItem(mi: MenuItem)

freeItems 0..n

MenuItem
description: String
+create(rec: Recipe, desc?: St
+create(mi: MenuItem)

sectionItems 0..n

MenuEventReceiver
{interface}
+*updateSectionAdded(m: Menu, sec: Section)*
+*updateMenuItemAdded(m: Menu, mi: MenuItem)*
+*updateMenuCreated(m: Menu)*
+*updateMenuDeleted(m: Menu)*
+*updateSectionsRearranged(m: Menu)*

General module

UseCaseLogicException
Exception

UserManagement

User
+isChef( ): boolean
+isCook(): boolean
+isPlanner(): boolean
+isWaitingStuff(): boolean
+giveAvailability(shift: Shift, recurrent: boolean)

1
0..1
0..n
0..1

0..1 currentUser

UserManager
event sender methods
operations methods
+getCurrentUser(): User

Detailed Sequence Diagram Cucina

[Kitchen] 1 generateSummarySheet

UI

sumshMngr:
SummarySheetManager

userMngr:
UserManager

generateSummarySheet(ev: Evento, serv: Service)

getCurrentUser()

user

ALT SumshMngr.user.isChef() && ev != null && ev.hasService(serv) && serv != null

generateSumSheet(ev: Evento, serv:Service)

sumSheet:
Summary Sheet

SumSheetReceiver:
SummarySheetReceiver

sumSheet.event = ev

sumSheet.service = service

sumSheet.owner = user

create

sumSheet.currentMiscList:
ArrayList<KitchenJob>

create

sumSheet.currentMiscList:
ArrayList<Task>

sumSheet

setCurrentSumSheet(sumSheet)

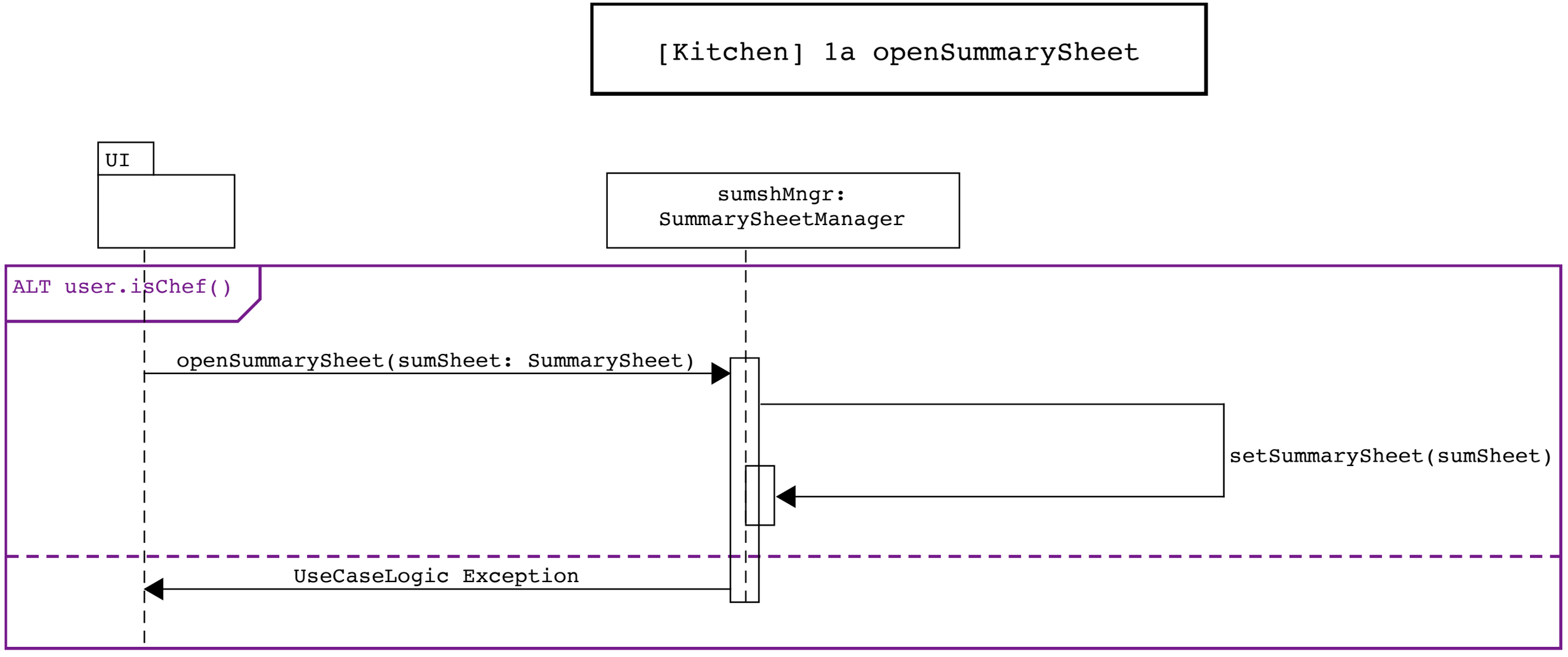
notifySummaryGenerated(sumSh: SummarySheet)

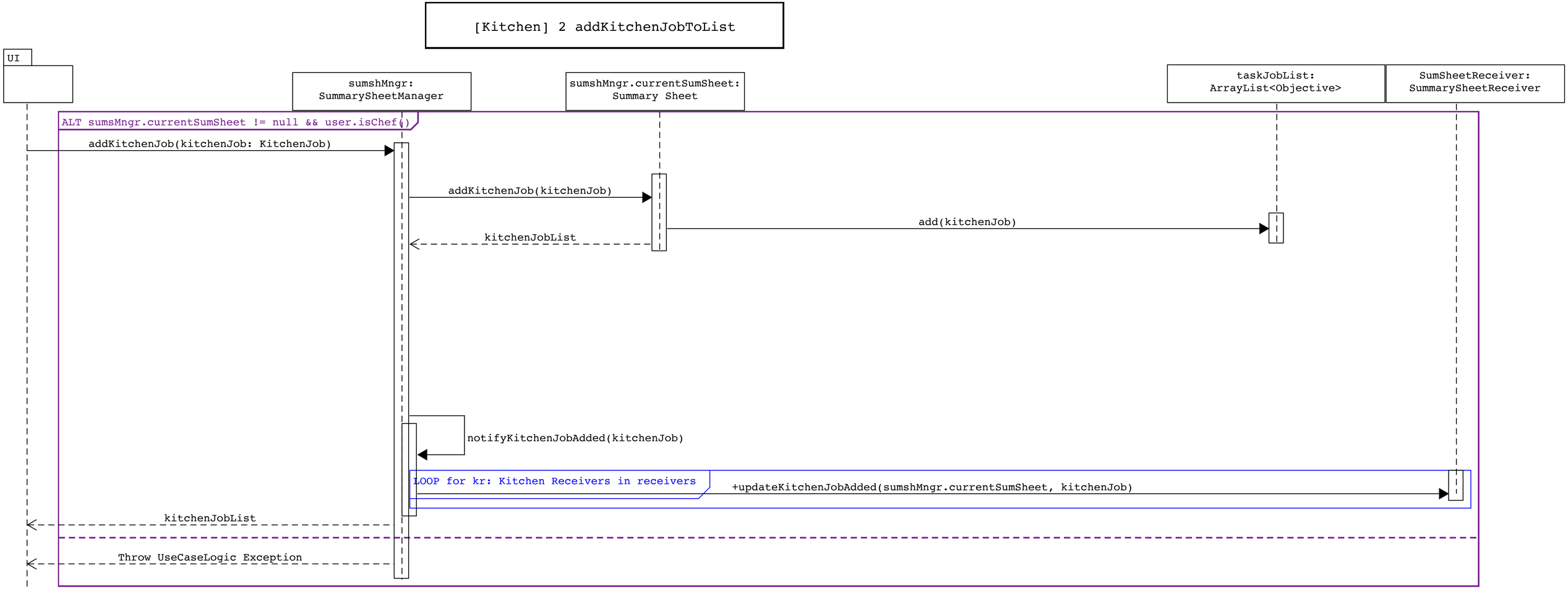
LOOP for kr: Kitchen Receivers in receivers

SummaryGenerated(sumSh: SummarySheet)

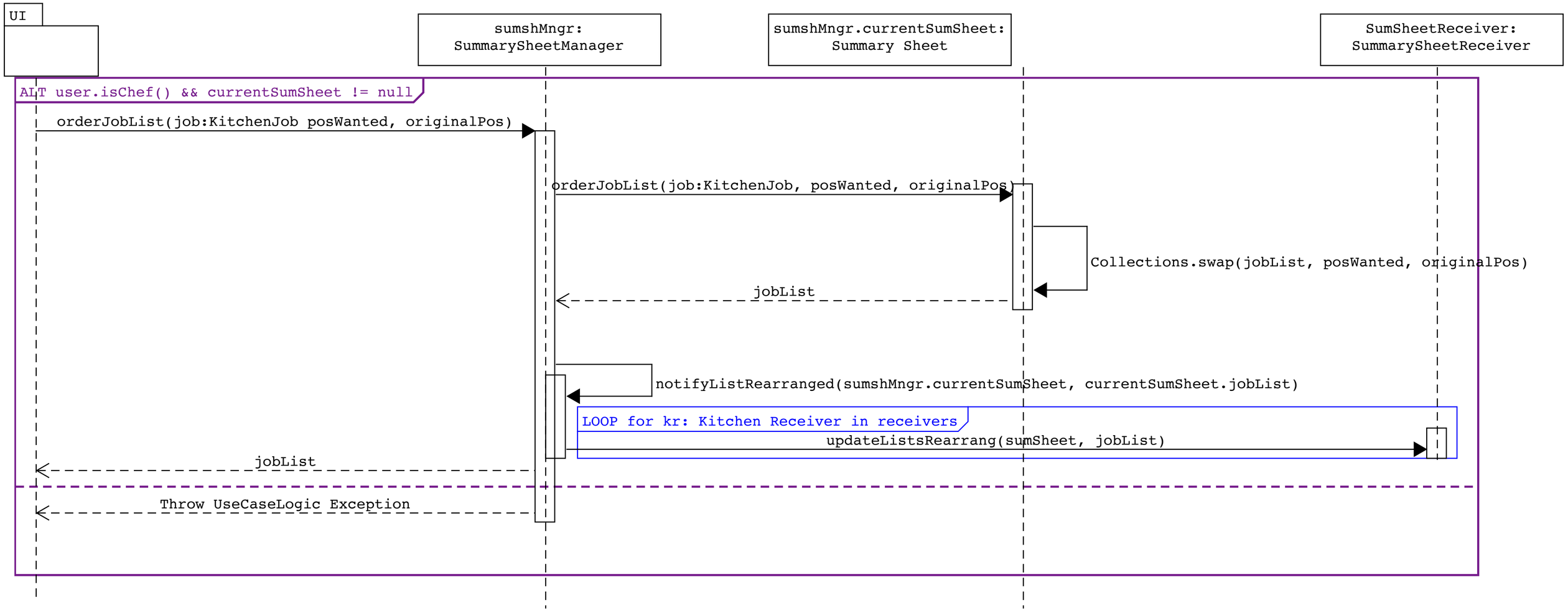
sumSheet

Throw UseCaseLogic Exception

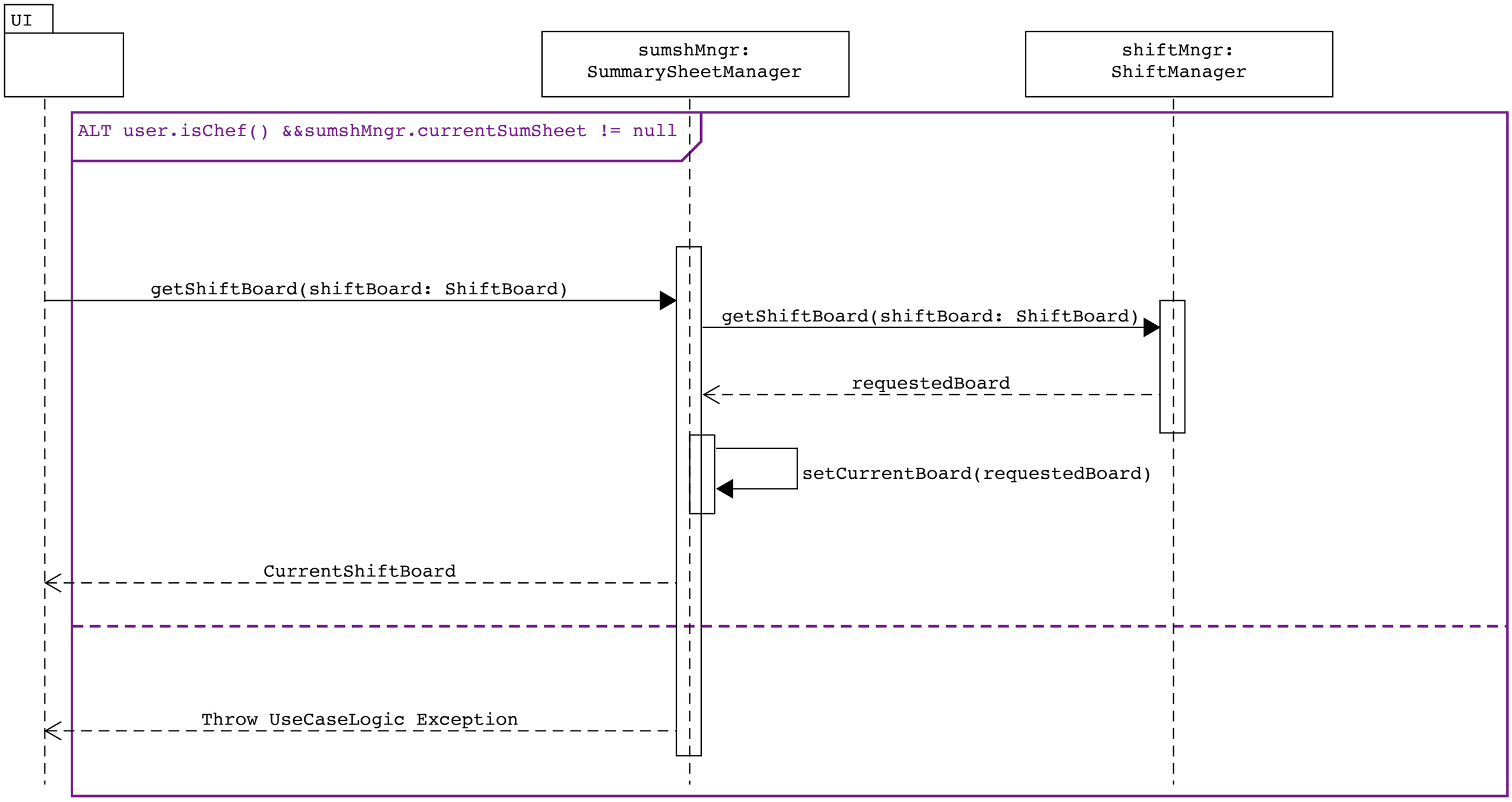

[Kitchen] 1a openSummarySheet
UI
sumshMngr:
SummarySheetManager
ALT user.isChef()
openSummarySheet(sumSheet: SummarySheet)
setSummarySheet(sumSheet)
UseCaseLogic Exception


[Kitchen] 2 addKitchenJobToList
UI
sumshMngr:
SummarySheetManager
sumshMngr.currentSumSheet:
Summary Sheet
taskJobList:
ArrayList<Objective>
SumSheetReceiver:
SummarySheetReceiver
ALT sumsMngr.currentSumSheet != null && user.isChef()
addKitchenJob(kitchenJob: KitchenJob)
addKitchenJob(kitchenJob)
add(kitchenJob)
kitchenJobList
notifyKitchenJobAdded(kitchenJob)
LOOP for kr: Kitchen Receivers in receivers
+updateKitchenJobAdded(sumshMngr.currentSumSheet, kitchenJob)
kitchenJobList
Throw UseCaseLogic Exception

[Kitchen] 3 orderJobList



[Kitchen] 4 getShiftBoard



[Kitchen] 5 createTask

UI

sumshMngr:
SummarySheetManager

sumshMngr.currentSumSheet:
Summary Sheet

currentSumSheet.taskList:
ArrayList<Task>

SumSheetReceiver:
SummarySheetReceiver

ALT currentSumSheet != null & user.isChef()

addTask(kitchenJob: KitchenJob, kitSh: Shift, coo?: Cook)

addTask(kitchenJob: KitchenJob, kitSh: Shift, coo?: Cook)

create(kitchenJob: KitchenJob, kitSh: Shift, coo?: Cook)

taskToAdd:
Task

ALT coo != null && kitSh.isAvailable(coo)

cook = coo

shift = kitSh

job= kitchenJob

newTask

add(newTask)

newTask

newTask

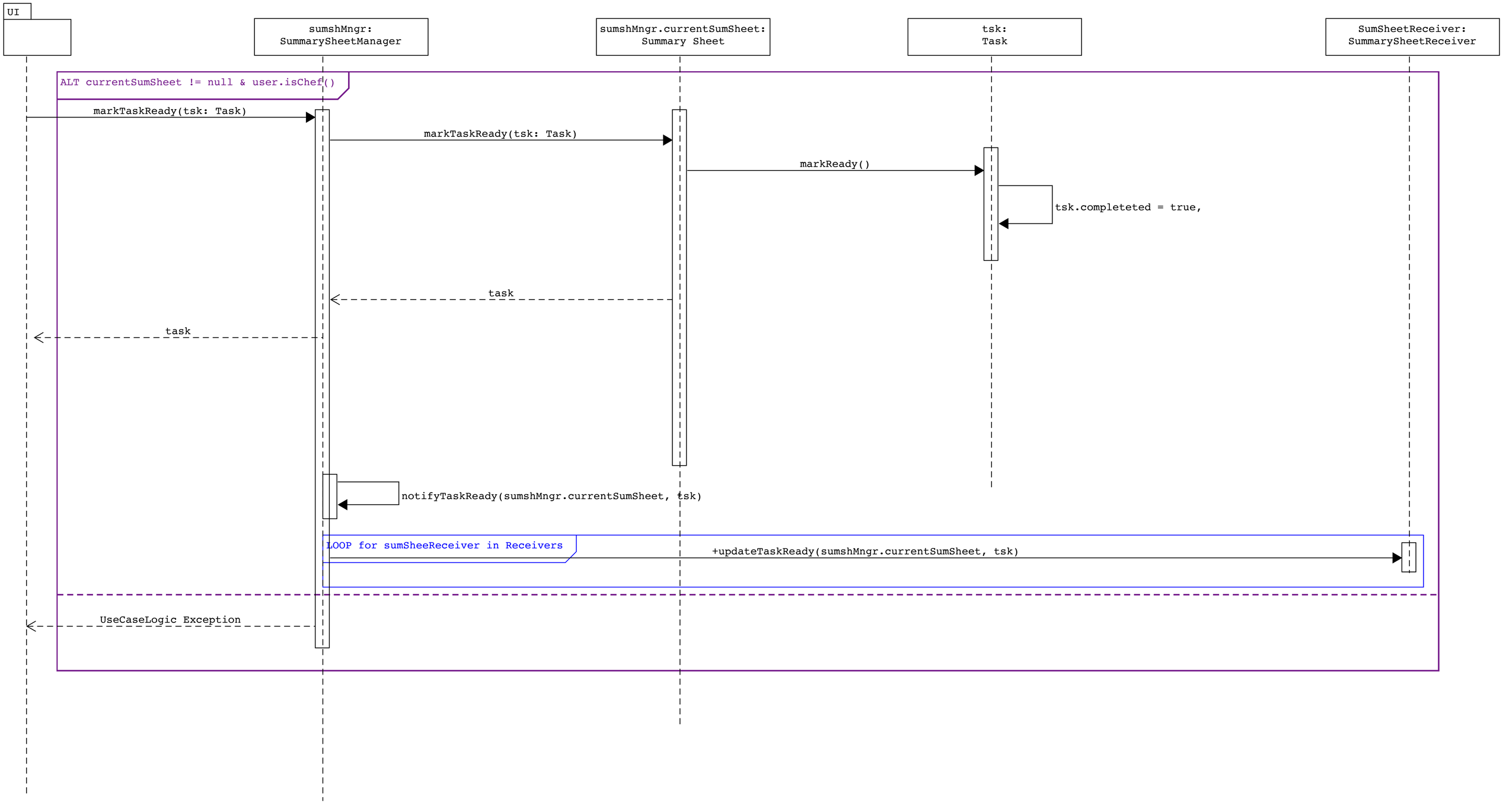
notifyTaskCreated(sumSh,newTask)

Throw UseCaseLogic Exception

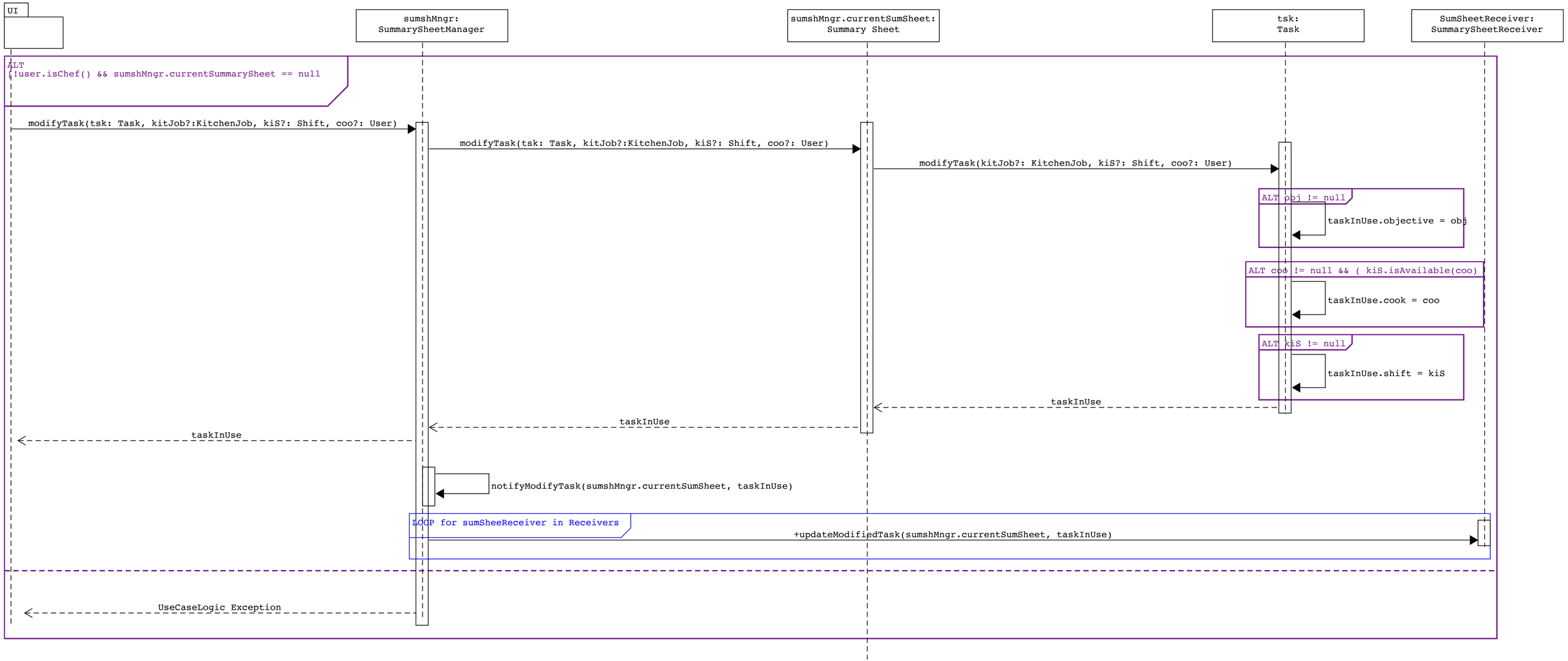
LOOP for sumSheeReceiver in Receivers

updateTaskCreated(tsk, sumSheet)

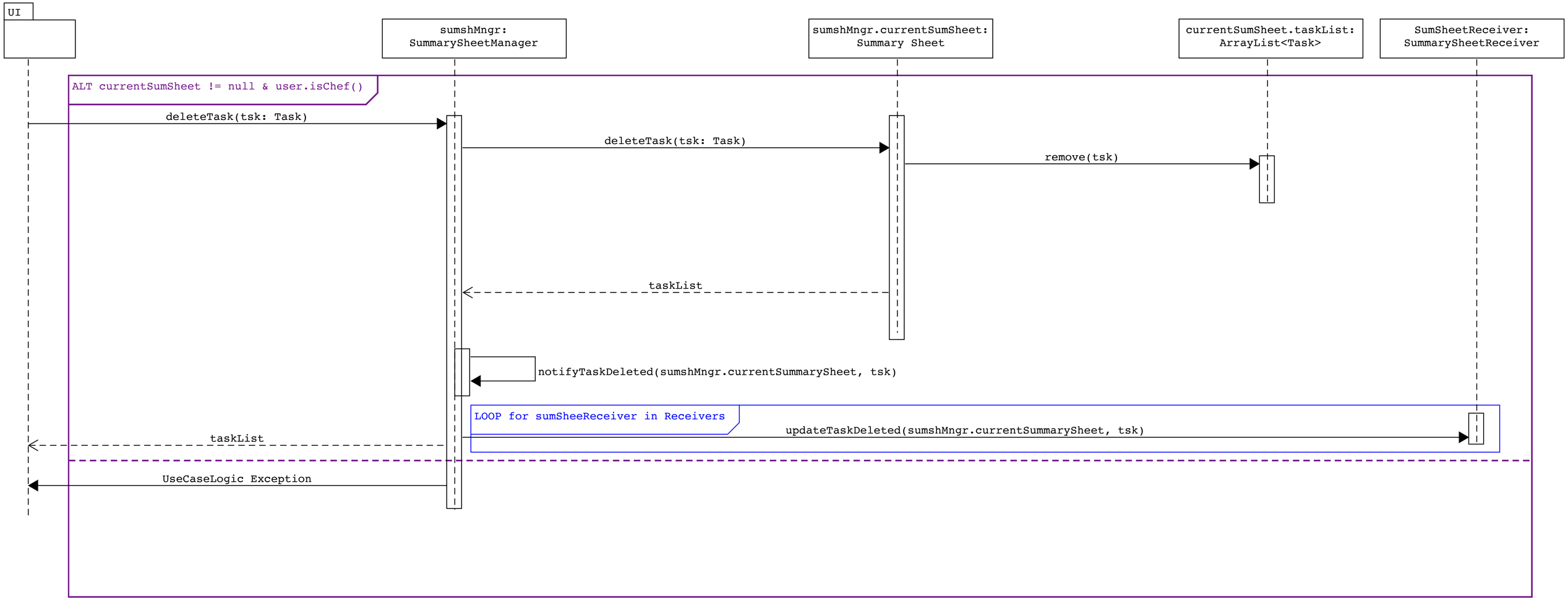
[Kitchen] 5a markTaskReady



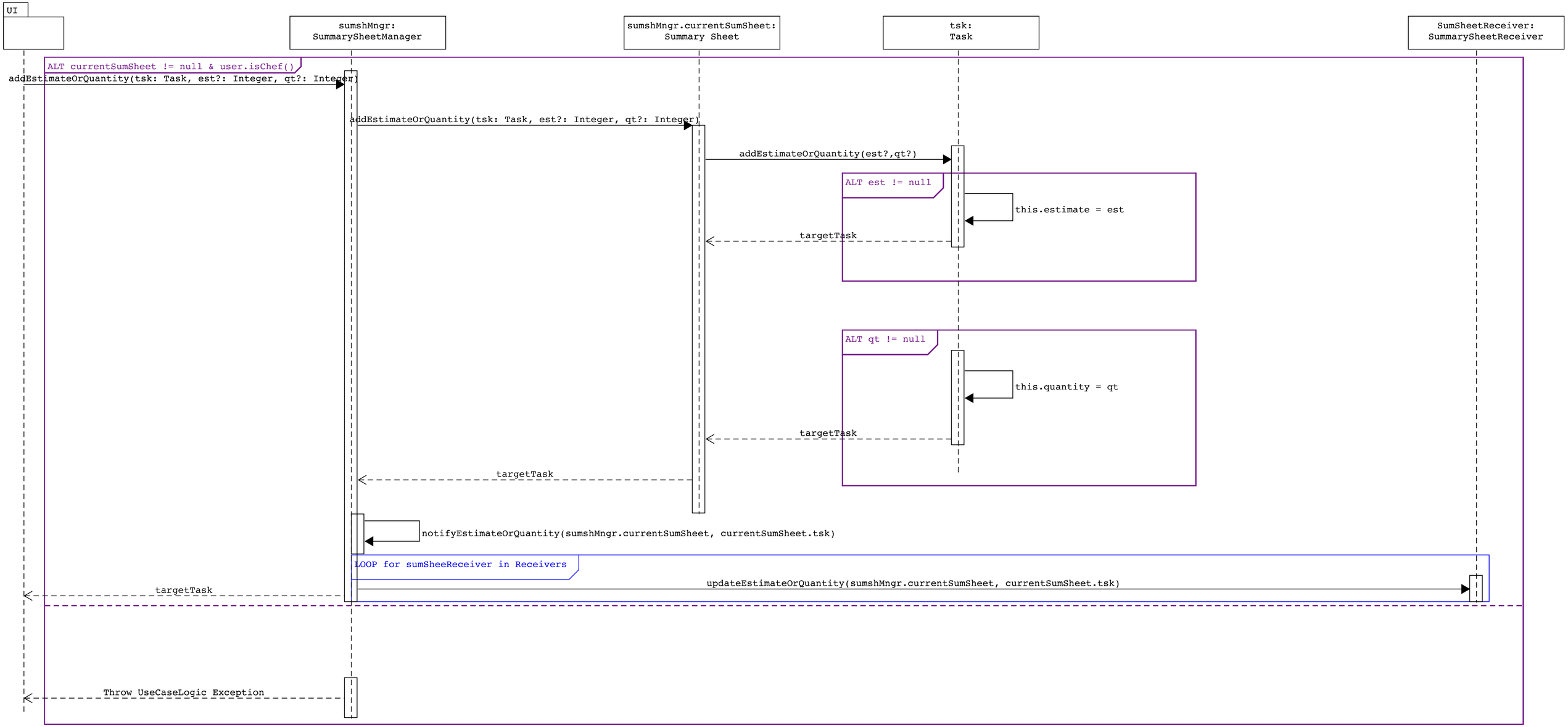
[Kitchen] 5b modifyTask



[Kitchen] 5c deleteTask



[Kitchen] 6 addEstimateOrQuantity



Sviluppo Applicazioni Software 20/21

Gianluca Cognigni

748049

Uc Dettagliati

Gestione turni

Informazioni generali

Nome caso d'uso: Gestione UC
Portata:
Livello:
Attore primario: Organizzatore
Parti Interessate: Cuochi, personale di sala
Pre-condizioni:
Garanzie di successo o post-condizioni: Aver dei compiti assegnati a turni per la preparazione dell'evento, (opzionale) anche dei cuochi assegnati ai vari compiti nei turni

Scenario principale di successo

#	Attore	Sistema
	Opzionalmente salta allo step 8 o allo step 6	
1	Stabilisce un turno singolo preparatorio	Il sistema crea un turno singolo e lo associa al tabellone dei turni
2	Opzionalmente segna se il lavoro della cucina va fatto in sede, in loco o altrove	Il sistema registra la preferenza sul luogo di preparazione sul turno in questione
3	Opzionalmente chiedi ai cuochi la disponibilità	Il sistema restituisce il tabellone dei turni
4	*Opzionalmente verifica la sovrapposizione dei turni preparatori*	Il sistema verifica la sovrapposizione di turni preparatori
5	*Opzionalmente modifica un turno singolo*	Il sistema registra le modifiche ad un turno singolo preparatorio
	Opzionalmente ritorna allo step 1	
	Opzionalmente salta allo step 8	
6	Opzionalmente passa in rassegna un evento	Il sistema fornisce all'utente l'evento in questione
7	Crea turno singolo di servizio per un evento	Il sistema crea un turno singolo di servizio
8	Opzionalmente modifica un turno di servizio	Il sistema registra la modifica al turno di servizio

9	Aggiunge una data di scadenza per modificare alcuni turni	Il sistema registra la data di scadenza per modificare i turni di servizio in questione
	Opzionalmente torna allo step 2 o allo step 8	

Eccezione 1.1a

#	Attore	Sistema
1	Utente cerca di creare turni singoli per una data precedente alla corrente	Il sistema impedisce la creazione e mostra il problema
	Termina il caso d'uso	

Eccezione 5.1a

#	Attore	Sistema
1	Utente cerca di modificare turni singoli per una data precedente alla corrente	Il sistema impedisce la modifica e lancia un'eccezione
	Termina il caso d'uso	

Eccezione 5.1b

#	Attore	Sistema
1	Utente cerca di modificare turni singoli una volta che le disponibilità iniziano ad esser date	Il sistema impedisce la creazione e mostra il problema
	Termina il caso d'uso	

Eccezione 7.1a

#	Attore	Sistema
1	Utente cerca di creare un turno di servizio per il servizio di un evento già	Il sistema impedisce la creazione e lancia un'eccezione

	iniziato	
	Torna allo scenario principale	

Eccezione 7.1b

#	Attore	Sistema
1	Utente cerca di creare un turno di servizio per una data precedente all'evento	Il sistema impedisce la creazione e lancia un'eccezione
	Torna allo scenario principale	

Eccezione 8.1a

#	Attore	Sistema
1	Utente cerca di modificare un turno di servizio una volta che le disponibilità iniziano ad esser date	Il sistema impedisce la creazione e mostra il problema
	Termina il caso d'uso	

Estensione 1a

#	Attore	Sistema
1	stabilisce un blocco di turni preparatorii singoli	Il sistema crea un blocco di turni di Cucina singoli
	Torna allo scenario principale	

Eccezione 1a.1a

#	Attore	Sistema
1	Cerca di stabilire un blocco di turni preparatorii precedenti alla data corrente	Il sistema impedisce la creazione e lancia un'eccezione

Estensione 1b

#	Attore	Sistema
1	stabilisce un turno preparatorio singolo ricorrente	Il sistema crea il turno singolo ricorrente e lo associa al tabellone dei turni
	Torna allo scenario principale	

Eccezione 1b.1a

#	Attore	Sistema
1	Cerca di stabilire un turno singolo preparatorio ricorrente precedente alla data corrente	Il sistema impedisce la creazione e lancia un'eccezione

Estensione 1c

#	Attore	Sistema
1	stabilisce blocco ricorrente di turni preparatorii ricorrenti	Il sistema crea blocchi ricorrenti di turni di Cucina
	Torna allo scenario principale	

Eccezione 1c.1a

#	Attore	Sistema
1	L'utente cerca di creare blocco ricorrente di turni preparatorii non ricorrenti	Il sistema impedisce la creazione ed avverte l'utente dell'eccezione
	Torna all'estensione 2d	

Eccezione 1c.1b

#	Attore	Sistema
1	Utente cerca di creare blocchi di turni ricorrenti prima della data corrente	Il sistema impedisce la creazione e lancia un'eccezione
	Torna allo scenario principale	

Estensione 5a

#	Attore	Sistema
1	Opzionalmente modifica il blocco di turni preparatorii	Il sistema registra le modifiche al blocco di turni preparatorii
	Torna allo scenario principale	

Eccezione 5a.1a

#	Attore	Sistema
1	Cerca di modificare un blocco di turni su cui è già stata data una disponibilità	Il sistema impedisce la modifica del blocco di turni e lancia un'eccezione
	Torna all'estensione 5a	

Eccezione 5a.1b

#	Attore	Sistema
1	Utente cerca di cerca di modificare i turni per una data precedente alla corrente	Il sistema impedisce la creazione e lancia un'eccezione
	Torna allo scenario principale	

Estensione 5b

#	Attore	Sistema
1	Opzionalmente elimina uno o più turni	Il sistema elimina uno o più turni
	Torna allo scenario principale	

Eccezione 5b.1a

#	Attore	Sistema
1	Cerca di eliminare uno o più turni su cui è già stata data una disponibilità	Il sistema impedisce l'eliminazione e lancia un'eccezione
	Torna all'estensione 5b	

Eccezione 5b.1b

#	Attore	Sistema
1	Utente cerca di eliminare uno o più turni di servizio per una data precedente all'evento	Il sistema impedisce la l'eliminazione e lancia un'eccezione
	Torna all'estensione 5b	

Estensione 7a

#	Attore	Sistema
1	Opzionalmente crea un turno ricorrente per un evento	Il sistema crea i turni e li fa ricorrere per il limite indicato
	Torna allo scenario principale	

Eccezione 7a.1a

#	Attore	Sistema

1	Cerca di creare turni ricorrenti per un evento ad una data precedente da quella dell'evento	Il sistema impedisce la creazione e lancia un'eccezione

Estensione 9a

#	Attore	Sistema
1	Opzionalmente aggiunge una data di scadenza per modificare tutti i turni	Il sistema registra la data di scadenza per modificare i turni
	Torna allo scenario principale	

Estensione 9b

#	Attore	Sistema
1	Opzionalmente aggiunge una data di scadenza per modificare un blocco	Il sistema registra la data di scadenza per modificare i turni
	Torna allo scenario principale	

Estensione 9c

#	Attore	Sistema
1	Opzionalmente elimina una scadenza per bloccare uno o più turni	Il sistema deregistra la scadenza per bloccare uno o più turni
	Torna allo scenario principale	

Estensione 9d

#	Attore	Sistema

1	Opzionalmente elimina una scadenza per bloccare tutti i turni	Il sistema deregistra la scadenza per bloccare tutti i turni
	Torna allo scenario principale	

Disponibilità

Informazioni generali

Nome caso d'uso: Gestione UC
Portata:
Livello:
Attore primario: Cuochi, personale di servizio
Parti Interessate: Cuochi, personale di sala, Organizzatori, Chef
Pre-condizioni:
Garanzie di successo o post-condizioni: Aver dei compiti assegnati a turni per la preparazione dell'evento, (opzionale) anche dei cuochi assegnati ai vari compiti nei turni

Scenario principale di successo

#	Attore	Sistema
	Opzionalmente salta allo step 2	
1	Da la disponibilità a turno singolo o di servizio o preparatorio	Il sistema registra la disponibilità
2	Opzionalmente elimina la disponibilità per un turno singolo o di servizio o preparatorio	Il sistema deregistra la disponibilità
	Opzionalmente torna allo step 1	

Eccezioni 1.1a

#	Attore	Sistema
1	Cerca di dare la disponibilità ad una data precedente della data corrente	Il sistema impedisce che ciò avvenga
	Termina il caso d'uso	

Eccezioni 1.1b - 2.1a

#	Attore	Sistema
1	Cerca di dare o ritirare la disponibilità temporalmente oltre ad una scadenza	Il sistema declina il registro della disponibilità per la scadenza sorpassata

	fissata dagli organizzatori	
	Termina il caso d'uso	

Estensione 1a

#	Attore	Sistema
1	Da la disponibilità per un blocco di Turni Preparatorio	Il sistema registra le disponibilità
	Torna allo scenario principale	

Eccezioni 1a.1a - 1b.1a - 1c.1a

#	Attore	Sistema
1	Cerca di dare la disponibilità o ritirarla temporalmente oltre ad una scadenza fissata dagli organizzatori	Il sistema declina il registro della disponibilità per la scadenza sorpassata
	Termina il caso d'uso	

Eccezioni 1a.1b - 1b.1b - 1c.1b

#	Attore	Sistema
1	Cerca di dare la disponibilità ad una data precedente della data corrente	Il sistema impedisce che ciò avvenga
	Termina il caso d'uso	

Eccezioni 1a.1c

#	Attore	Sistema
1	Cerca di dare la disponibilità in blocco ad un turno di servizio	Il sistema impedisce che ciò avvenga
	Termina il caso d'uso	

Estensione 1b

#	Attore	Sistema
1	Opzionalmente dà la disponibilità per turni singoli ricorrenti	Il sistema registra le disponibilità
	Torna allo scenario principale	

Eccezione 1b.1c

#	Attore	Sistema
1	Un cuoco cerca di dare la disponibilità violando la direzione dello chef di dare la disponibilità ad un blocco	Il sistema declina il registro della disponibilità
	Termina il caso d'uso	

Estensione 1c

#	Attore	Sistema
1	Opzionalmente dà la disponibilità per blocchi ricorrenti di turni preparatorii	Il sistema registra le disponibilità
	Torna allo scenario principale	

Eccezioni 1c.1c

#	Attore	Sistema
1	Cerca di dare la disponibilità in blocchi ricorrenti a dei turni di servizio	Il sistema impedisce che ciò avvenga
	Termina il caso d'uso	

Estensione 2a

#	Attore	Sistema
1	Opzionalmente elimina la disponibilità	Il sistema elimina le disponibilità al turno

	per un blocco di turni preparatorii	
	Torna allo scenario principale	

Estensione 2b

#	Attore	Sistema
1	Opzionalmente elimina la disponibilità per blocchi ricorrenti di turni preparatorii	Il sistema elimina le disponibilità dal turno
	Torna allo scenario principale	

Estensione 2c

#	Attore	Sistema
1	Opzionalmente elimina la disponibilità per turni singoli ricorrenti o di servizio o preparatorii	Il sistema elimina le disponibilità dal turno
	Torna allo scenario principale	

Eccezioni 2a.1a - 2b.1a - 2c.1a

#	Attore	Sistema
1	Cerca di ritirare la disponibilità temporalmente oltre ad una scadenza fissata dagli organizzatori	Il sistema declina il registro della disponibilità per la scadenza sorpassata
	Termina il caso d'uso	

Sezione
nome: testo

Una voce puo' comparire
menu' all'interno di una
oppure come voce "libera"
verrà valorizzata una e
delle due relazioni app

0..n
contiene
1
0..1
appartiene
0..n

Menù
titolo: testo
pubblicato: sì/no
richiede cuoco: sì/no
piatti caldi: sì/no
richiede cucina: sì/no

0..1
appartiene
0..n

Voce
descrizione: tes

in uso in
0..1
0..n
0..n
fa riferimento a
1

Ricetta
nome: testo
autore: testo
descrizione: testo
numero porzioni: n
note: testo
tag: testo

si trova in
0..n
1
0..n
contenuto
0..n

0..n
1..n
Evento
nome: testo
informazioni: te
ricorrente: si/n

Service
name: testo

prevede
1..n
1
1
1
associato a
0..1
0..n
assegnato a
0..1

Preparazione

0..n
0..1
contiene
1..n
si trova in

Ingrediente
nome: testo
quantità: numero

Foglio Riepigolativo
titolo: nome
informazioni: testo

creato da
0..n 0..1

Chef
nome: testo

ha in gestione
associato a
1
0..1
0..1
contiene {Ordered}
0..n
crea
contiene
contiene info uguali
1
0..n
1
0..n
si trova in
1
Ricettario

Mansione

Se l'attributo "completato" di un dato compito
è impostato a si, allora la mansione è collegata
per forza ad un turno ed un cuoco

Organizzatore
nome: testo

1
1
0..1
1
1
consiste in
0..1
0..n
si trova in
1
ElencoIngredienti

Tabellone Turni
/scadenzaGlobale: Data
currentDate: testo

Compito
quantità: numero
completato: si/n
stima: numero

0..n
Cuoco
nome: testo

0..n
eseguito
0..1

contiene
0..n
eseguita in
0..n
0..1
lavora in
0..n
0..n
gestisce

Turno Preparatorio
/parteDiBlocco: si/no

Turno
/data: Data
fasciaOraria: numero
/scadenza: Data
posizioneLavori: testo

0..n
coordina
da disponibilita
0..n 0..n

Personale di Servizio

0..n
raggruppati in
0..1

Blocco

coordina
1
1
0..n
0..n
coordina

Personale di Sala
nome: testo

Calendario
dataCorrente: Data

possiede

Turno di servizio
tempoAggiuntivoPrima: numero
tempoAggiuntivoDopo: numero

1
coordina
0..1
0..1

Ricorrenza
/dataInizio: Data
/dataFine: Data
cadenza: {settimanale, mensile, annuale}

SSD Turni

Diagramma di Sequenza di Sistema per UC "Turni" Scenario Principale

Organizzatore

Sistema

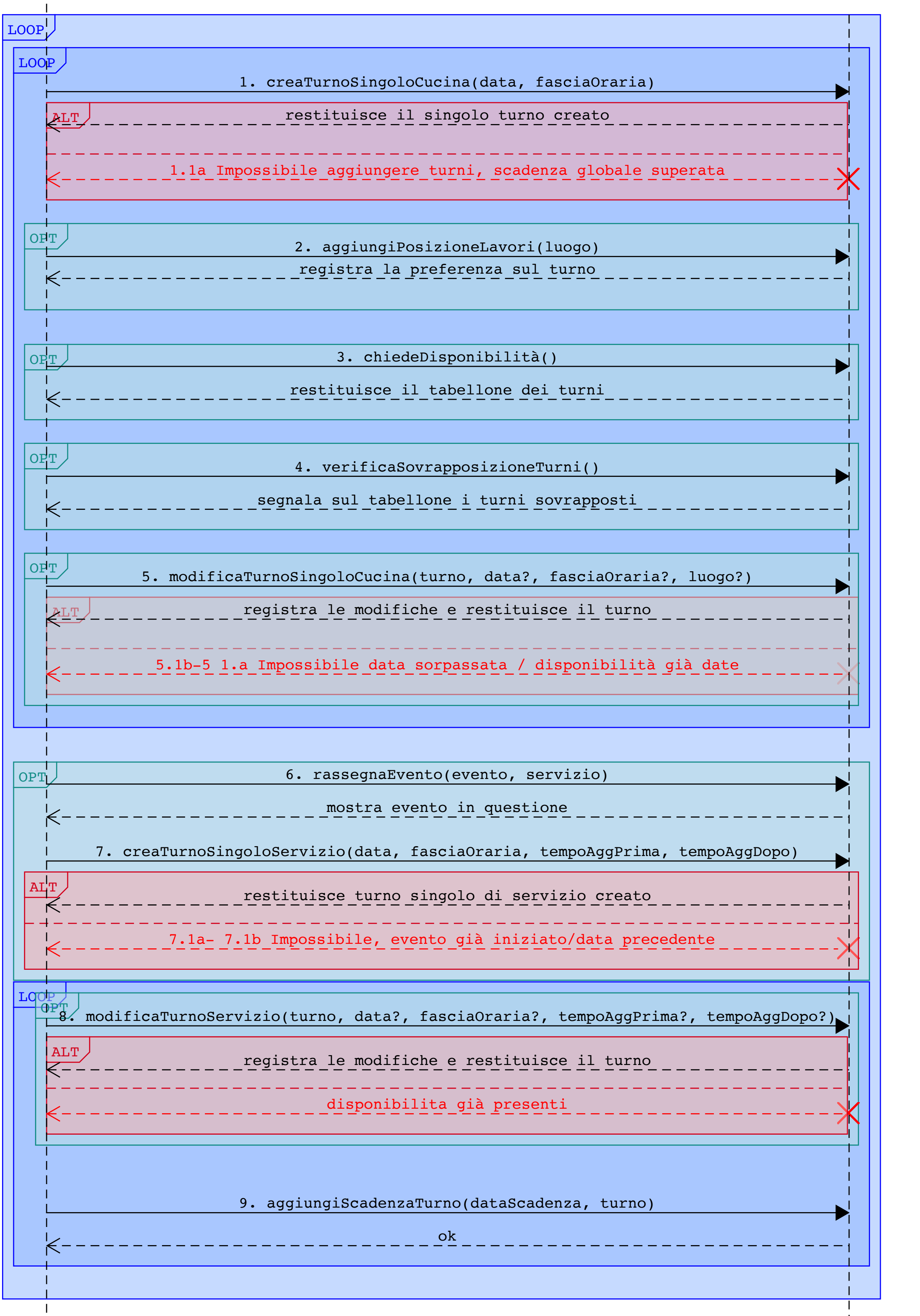
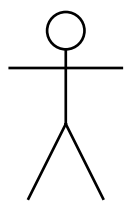


Diagramma di Sequenza di Sistema per UC "Gestione Turni" Estensione 2a-2b-2c



Organizzatore

Sistema

LOOP

LOOP

ALT

1. creaTurnoPrepSingolo(data, fasciaOraria)

ok

1a creaBloccoTurniPrep(turni)

ALT

ok

2a.1a eccezione: scadenza sorpassata

1b creaTurnoPrepRicorrente(turno, cadenza, dataInizio, dataFine)

ALT

ok

2b.1a eccezione: scadenza sorpassata

2c creaBloccoPrepRicorrente(turniRicorrenti, cadenza, dataInizio, dataFine)

ALT

ok

2c.1a eccezione: scadenza sorpassata

2c.1b eccezione: turni non ricorrenti

OPT

2. aggiungiPosizioneLavori(posizione)

registra la preferenza

OPT

3. chiedeDisponibilità()

restituisce il tabellone dei turni

OPT

4. verificaSovrapposizioneTurni()

segnala sul tabellone i turni sovrapposti

OPT

ALT

5. modificaTurnoSingoloCucina(turno, data?, fasciaOraria?, luogo?)

registra le modifiche e restituisce il turno

5.1b-5 1.a Impossibile data sorpassata / disponibilità già date

OPT

6. rassegnaEvento(evento, servizio)

mostra evento

7. creaTurnoSingoloServizio(data, fasciaOraria, tempoAggPrima, tempoAggDopo)

ALT

restituisce turno singolo di servizio creato

7.1a- 7.1b Impossibile, evento già iniziato/data precedente

LOOP

OPT

8. modificaTurnoServizio(turno, data?, fasciaOraria?, tempoAggPrima?, tempoAggDopo?)

ALT

registra le modifiche e restituisce il turno

disponibilita già presenti

9. aggiungiScadenzaTurni(dataScadenza, turni)

ok

Diagramma di Sequenza di Sistema per UC
"Turni"
Estensione 5a-5b

Organizzatore

Sistema

LOOP
LOOP
ALT
1. creaTurnoSingoloCucina(data, ora)
restituisce il singolo turno creato
1.1a Impossibile aggiungere turni, data superata
OPT
2. aggiungiPosizioneLavori(posizione)
registra la preferenza
OPT
3. chiedeDisponibilità()
restituisce il tabellone dei turni
OPT
4. verificaSovrapposizioneTurni()
segnala sul tabellone i turni sovrapposti
ALT
OPT 5. modificaTurnoSingoloCucina(turnoPrep, data?, fasciaOraria?, luogo?)
ALT
ok
5.1a eccezione: data precedente odierna
5.1b eccezione: disponibilità già presenti
OPT 5a. modificaBloccoCucina(blocco, data?, fasciaOraria?, luogo?)
ALT
ok
5a.1a eccezione: disponibilità già presenti
5b.1a eccezione: data precedente odierna
OPT
5b eliminaTurno(turno?, blocco?)
ALT
ok
5b.1a eccezione: disponibilità già presenti
5b.2a eccezione: data precedente odierna
OPT
6. rassegnaEvento(evento, servizio)
mostra evento
7. creaTurnoSingoloServizio(data, fasciaOraria, tempoAggPrima, tempoAggDopo)
ALT
restituisce turno singolo di servizio creato
7.1a- 7.1b Impossibile, evento già iniziato/data precedente
LOOP
OPT
8. modificaTurnoServizio(turno, data?, fasciaOraria?, tempoAggPrima?, tempoAggDopo?)
ALT
registra le modifiche e restituisce il turno
disponibilita già presenti
9. aggiungiScadenzaTurni(dataScadenza, turni)
ok

Diagramma di Sequenza di Sistema per UC "Turni" Estensione 7a

Organizzatore

Sistema

LOOP

LOOP

1. creaTurnoSingoloCucina(data, fasciaOraria)

ALT

restituisce il singolo turno creato

1.1a Impossibile aggiungere turni, data superata

OPT

2. aggiungiPosizioneLavori(posizione)

registra la preferenza sul turno

OPT

3. chiedeDisponibilità()

restituisce il tabellone con i turni

OPT

4. verificaSovrapposizioneTurni()

segnala sul tabellone i turni sovrapposti

OPT

5. modificaTurnoSingoloCucina(turno, data?, fasciaOraria?, luogo?)

ALT

registra le modifiche e restituisce il turno

5.1b-5 1.a Impossibile data sorpassata / disponibilità già date

OPT

6. rassegnaEvento(evento, servizio)

mostra evento

ALT

7. creaTurnoSingoloServizio(data, fasciaOraria, tempoAggPrima, tempoAggDopo)

ALT

restituisce il turno di servizio creato

7.1a servizio già iniziato

OPT

7a creaTurnoServizioRicorrente(turnoServizio, cadenza, dataInizio, dataFine)

ALT

restituisce i turni di servizio creati

7b.1a eccezione: data precedente odierna non disponibile

LOOP

OPT

8. modificaTurnoServizio(turno, data?, fasciaOraria?, tempoAggPrima?, tempoAggDopo?)

ALT

registra le modifiche e restituisce il turno

disponibilita già presenti

9. aggiungiScadenzaTurni(dataScadenza, turni)

ok

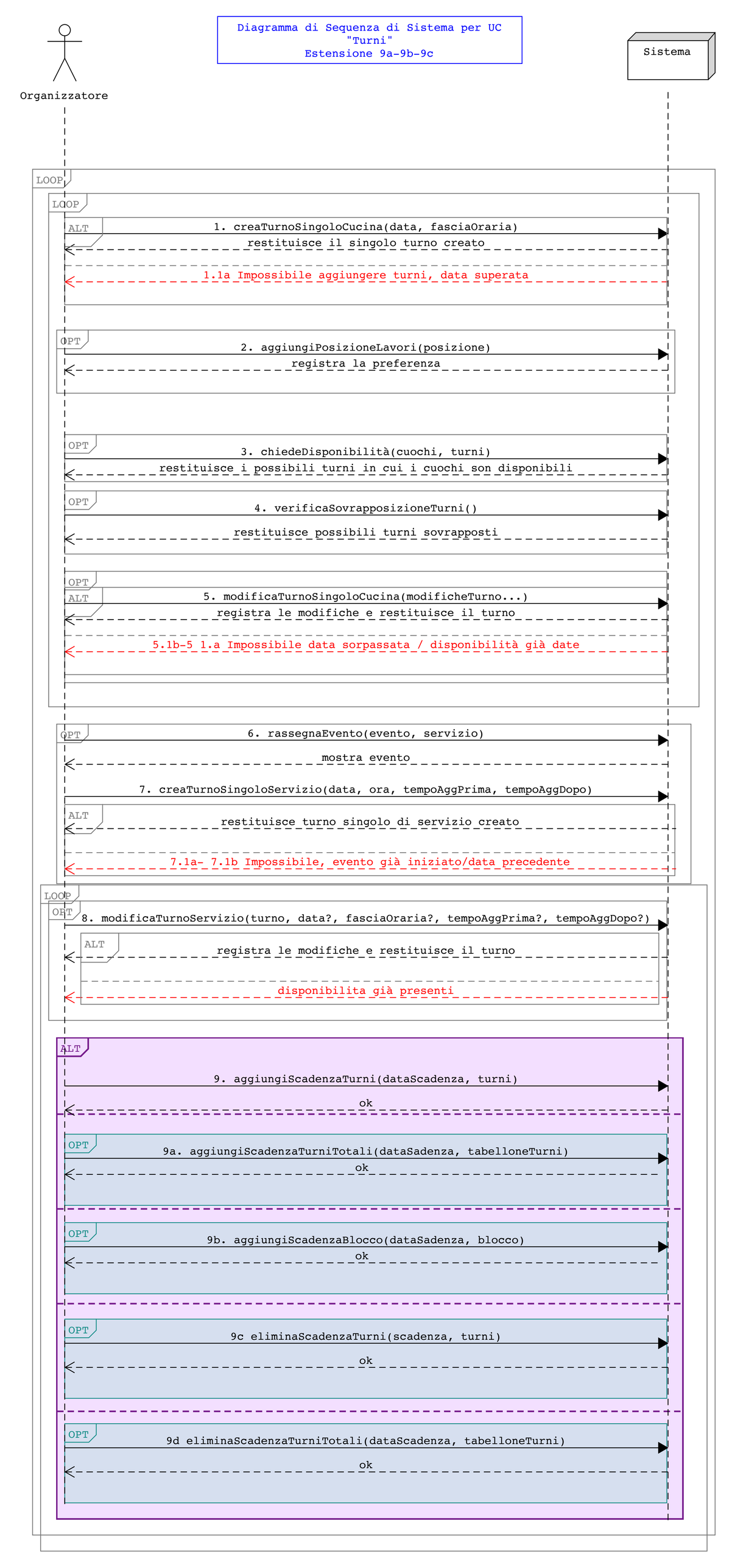


Diagramma di Sequenza di Sistema per UC
"Turni"
Estensione 9a-9b-9c
Organizzatore
Sistema
LOOP
LOOP
ALT
1. creaTurnoSingoloCucina(data, fasciaOraria)
restituisce il singolo turno creato
1.1a Impossibile aggiungere turni, data superata
OPT
2. aggiungiPosizioneLavori(posizione)
registra la preferenza
OPT
3. chiedeDisponibilità(cuochi, turni)
restituisce i possibili turni in cui i cuochi son disponibili
OPT
4. verificaSovrapposizioneTurni()
restituisce possibili turni sovrapposti
OPT
ALT
5. modificaTurnoSingoloCucina(modificheTurno...)
registra le modifiche e restituisce il turno
5.1b-5 1.a Impossibile data sorpassata / disponibilità già date
OPT
6. rassegnaEvento(evento, servizio)
mostra evento
7. creaTurnoSingoloServizio(data, ora, tempoAggPrima, tempoAggDopo)
ALT
restituisce turno singolo di servizio creato
7.1a- 7.1b Impossibile, evento già iniziato/data precedente
LOOP
OPT
8. modificaTurnoServizio(turno, data?, fasciaOraria?, tempoAggPrima?, tempoAggDopo?)
ALT
registra le modifiche e restituisce il turno
disponibilita già presenti
ALT
9. aggiungiScadenzaTurni(dataScadenza, turni)
ok
OPT
9a. aggiungiScadenzaTurniTotali(dataSadenza, tabelloneTurni)
ok
OPT
9b. aggiungiScadenzaBlocco(dataSadenza, blocco)
ok
OPT
9c eliminaScadenzaTurni(scadenza, turni)
ok
OPT
9d eliminaScadenzaTurniTotali(dataScadenza, tabelloneTurni)
ok

SSD Disponibilità

Diagramma di Sequenza di Sistema per UC "Disponibilità" Scenario Principale

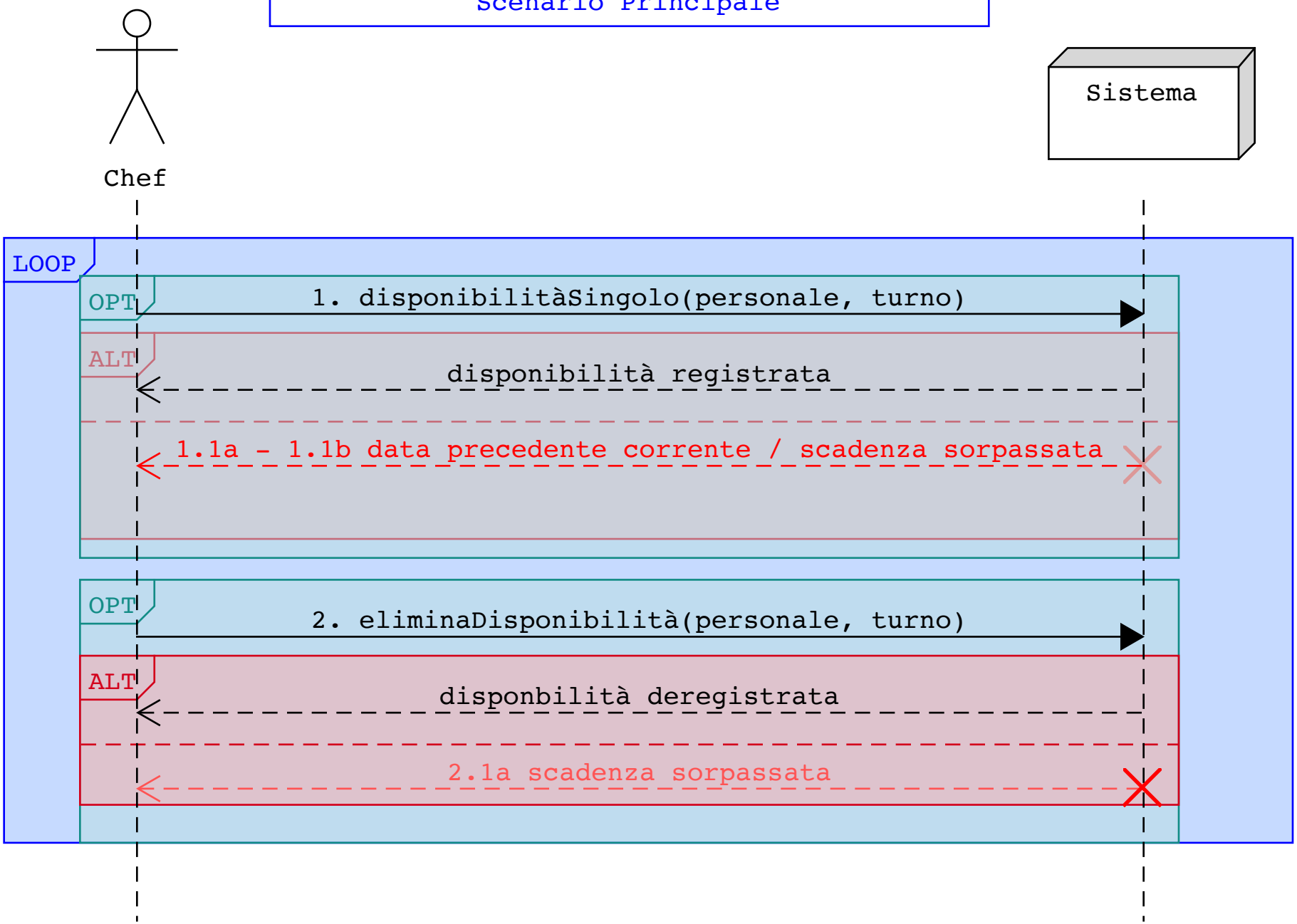
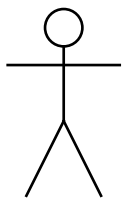


Diagramma di Sequenza di Sistema per UC
"Disponibilità"
Estensione 1a-1b-1c



Chef

Sistema

LOOP

ALT

1a disponibilitàBlocco(blocco, personale)

ALT

ok

1a.1a eccezione: scadenza sorpassata

1a.1b eccezione: eccezione: data non disponibile

OPT

1b disponibilitaSingolaRicorrente(turno, personale)

ALT

ok

1a.1a eccezione: scadenza sorpassata

1b.1b eccezione: eccezione: data non disponibile

1b.1c eccezione: disponibilita da dare in blo

OPT

1c disponibilitaBloccoRicorrente(blocco, personale)

ALT

ok

1a.1a eccezione: scadenza sorpassata

1c.1b eccezione: eccezione: data non disponibile

OPT

2. eliminaDisponibilità(personale, turno)

ALT

disponbilità deregistrata

2.1a scadenza sorpassata

Diagramma di Sequenza di Sistema per UC
"Disponibilità"
Estensione 2a-2b-2c

Chef

Sistema

LOOP

OPT

1. disponibilitàSingolo(personale, turno)

ALT

disponibilità registrata

1.1a - 1.1b data precedente corrente / scadenza sorpassata

ALT

OPT

2. eliminaDisponibilità(turno)

ALT

disponbilità deregistrata

2.1a eccezione: operazione non più possibile

OPT

2a eliminaBlocco(blocco)

ALT

ok

2a.1a eccezione: scadenza sorpassata

OPT

2b eliminaBloccoRicorrente(turnoBloRic)

ALT

ok

2b.1a eccezione: scadenza sorpassata

OPT

2c eliminaSingoloRicorrente(turnoRic)

ALT

ok

3c.1a eccezione: scadenza sorpassata

Contratti Turni

Scenario Principale

Condizioni generali

- L'attore è identificato con un'istanza *org* di Organizzatore
- Esistenza di un'istanza *tab* di Tabellone Turni
- Esistenza di un'istanza *cal* di Calendario
- *cal* **coordina** *tab*
- *tab.*scadenzaGlobale > a *cal.*dataCorrente

1. creaTurnoSingoloCucina(data: Data, fasciaOraria: numero)

pre-condizione:

- -

post-condizione:

- Viene creata un'istanza *tuP* di Turno Preparatorio
- *tuP.*data = data
- *tuP*.fasciaOraria = fasciaOraria
- *tab* **contiene** *tuP*
- *cal* **coordina** *tuP*

2. aggiungiPosizioneLavori(posizione: testo)

pre-condizione:

- è in corso la definizione di un Turno Preparatorio *tuP*

post-condizione:

- *tuP.*posizioneLavori = posizione

3. chiediDisponibilità()

pre-condizione:

- è in corso la definizione di un Turno Preparatorio *tuP*

post-condizione:

- è una interrogazione, non esistono post condizioni

4. verificaSovrapposizioneTurni()

pre-condizione:

- è in corso la definizione di un Turno Preparatorio *tuP*

post-condizione:

- è una interrogazione, non esistono post condizioni

5. modificaTurnoSingoloCucina(turnoPrep: Turno Preparatorio, data?: Data, fasciaOraria?: numero, luogo?: testo)

pre-condizione:

- *cal* **coordina** turnoPrep
- l'istanza *tab* di Tabellone turni **contiene** l'istanza *turnoPrep,*
- *turno.*scadenza > *tab*.dataCorrente,

post-condizione:

- (Se data specificato) turnoPrep.data = data
- (Se fasciaOraria specificato) *turnoPrep.*fasciaOraria = fasciaOraria
- (Se luogo specificato) turnoPrep.luogo = luogo

6. rassegnaEvento(evento: Evento, servizio: Servizio)

pre-condizione:

- Avere l'istanza evento in questione esistente
- Avere l'istanza servizio in questione esistente
- evento **prevede** servizio

post-condizione:

- é una interrogazione al sistema

7. creaTurnoSingoloServizio(evento: Evento, servizio: Servizio ,data: Data ,fasciaOraria: numero, tempoAggPrima: numero, tempoAggDopo: numero)

pre-condizione:

- evento **prevede** *servizio*

post-condizione:

- Creata un'istanza *tuS* di Turno di Servizio
- *tuS.*data = data
- *tuS*.ora = ora
- *tuS*.tempoAggiuntivoPrima = tempoAggPrima
- *tuS.*tempoAggiuntivoDopo = tempoAggDopo
- l'istanza *tab* di Tabellone Turni **contiene** *tuS*
- *cal* **coordina** *tuS*

8. modificaTurnoServizio(turnoServizio: Turno Servizio, data?: Data, fasciaOraria?: numero, luogo?: testo)

pre-condizione:

- *tab* **contiene** *turnoServizio*
- *cal* **coordina** turnoServizio
- *turnoServizio.scadenza* > *cal.*dataCorrente

post-condizione:

- (Se data specificato) *turno.data* = data
- (Se fasciaOraria specificato) *turno.*fasciaOraria = fasciaOraria
- (Se luogo specificato) *turno.luogo* = luogo

9. aggiungiScadenzaTurno(data: Data, turno: Turno)

pre-condizione:

- *tab* **contiene** turno
- l'attributo *tab.*scadenzaGlobale > data

post-condizione:

- L'attributo turno.scadenza = data

Estensione 1a-1b-1c

Condizioni generali

- L'attore è identificato con un'istanza *org* di Organizzatore
- Esistenza di un'istanza *tab* di Tabellone Turni
- Esistenza di un'istanza *cal* di Calendario
- *cal* **coordina** *tab*
- *tab.*scadenzaGlobale > a *cal.*dataCorrente

1a. creaBloccoTurniPrep(insieme T di Turni Preparatorii)

pre-condizione:

-

post-condizione:

- è stata creata un'istanza *bloc* di Blocco
- Per ogni turno *t* dell'insieme T, *t* ***raggruppato in*** *bloc*
- Per ogni turno *t* dell'insieme T l'attributo *t.*parteDiBlocco = si
- *bloc*.fasciaOraria = min(t.fasciaOraria) + max(t.fasciaOraria)
- *cal* **coordina** *bloc*
- *tab* **contiene** *bloc*

1b. creaTurnoPrepRicorrente(turnoPrep: TurnoPreparatorio, cadenza: enumerazione, dataInizio: numero, dataFine: numero)

pre-condizione:

post-condizione:

[se *tab* **contiene** turnoPrep]

- è stata creata un'istanza *ric* di Ricorrenza
- *cal* **coordina** *ric*
- turnoDiServizio **possiede** *ric*
- *ric*.cadenza = cadenza

- *ric.*dataInizio = dataInizio
- *ric.*dataFine = dataFine
- [se cadenza == settimanale]

  è stata creato un'elenco T di istanze *tuPN* di Turno Preparatorio con ogni istanza *tuP avente* attributo *tuP*.data = turnoPreparatorio.data + 7 rispetto a *tuP -1, ognuna* con dataInizio < *tuP.data* < *dataFine*
- [se cadenza == mensile]

  è stata creato un'elenco T di istanze *tuPN* di Turno Preparatorio con ogni istanza *tuP avente* attributo *tuP*.data = *turnoPrep.*data + 30 rispetto a *tuP -1, ognuna* con dataInizio < *tuS.data* < *dataFine*
- [se cadenza == annuale]

  è stata creato un'elenco T di istanze *tuPN* di Turno di Servizio con ogni istanza *tuP avente* attributo *tuS*.data = turnoPrep.data + 365 rispetto a *tuP -1, ognuna* con dataInizio < *tuP.data* < *dataFine*
- Ogni istanza *tuP* ***possiede*** *ric*
- Per ogni istanza *tuPN, tab* **possiede** *tuP*
- Per ogni istanza *tuPN*, *cal* **coordina** *tuP*

1c. creaBloccoPrepRicorrente(blocco: Blocco, cadenza: enumerazione, dataInizio: numero, dataFine: numero)

pre-condizione:

—

post-condizione:

[se *tab* **contiene** blocco]

- è stata creata un'istanza *ric* di Ricorrenza
- *cal* **coordina** *ric*
- blocco **possiede** *ric*
- *ric*.cadenza = cadenza
- *ric.*dataInizio = dataInizio
- *ric.*dataFine = dataFine

- [se cadenza == settimanale]

  è stata creato un'elenco T di istanze *bloc* di Blocco con ogni istanza *tuS avente* attributo *bloc*.data = *blocco.*data + 7 rispetto a *bloc -1, ognuna* con dataInizio < *bloc.data* < *dataFine*

- [se cadenza == mensile]

  è stata creato un'elenco T di istanze *bloc*N di Blocco con ogni istanza *bloc avente* attributo *bloc*.data = blocco.data + 30 rispetto a *bloc -1, ognuna* con dataInizio < *bloc.data* < *dataFine*

- [se cadenza == annuale]

  è stata creato un'elenco T di istanze *blocN* di Blocco con ogni istanza *bloc avente* attributo *bloc*.data = *blocco.*data + 365 rispetto a *bloc -1, ognuna* con dataInizio < *bloc.*data < *dataFine*

- Per ogni istanza *blocN, bloc* ***possiede*** *ric*
- Per ogni istanza *blocN, tab* **contiene** *bloc*
- Per ogni istanza *blocN, cal* **coordina** *bloc*

Estensione 5a-5b

Condizioni generali

- L'attore è identificato con un'istanza *org* di Organizzatore
- Esistenza di un'istanza *tab* di Tabellone Turni
- Esistenza di un'istanza *cal* di Calendario
- *cal* **coordina** *tab*
- *tab.*scadenzaGlobale > a *cal.*dataCorrente

5a. modificaBloccoTurniPrep(blocco: Blocco, data?: Data, fasciaOraria?: numero, luogo?: testo)

pre-condizioni:

- Non esista alcuna associazione *personale* (istanza di Personale di Servizio)
- blocco.scadenza > *cal.*dataCorrente

post-condizione:

- (Se data specificato) blocco*.data* = *data*
- (Se fasciaOraria specificato) blocco*.ora* = *fasciaOraria*
- (Se luogo specificato) blocco*.luogo* = *luogo*

5b. eliminaTurno(turno?: Turno, blocco?: Blocco)

pre-condizioni:

post-condizioni:

- [Se non esiste alcuna istanza di Personale di Servizio che **da disponibilità** a Turno]
  - [se turno specificato e turno.scadenza > *cal.*dataCorrente]

    tab **non contiene** turno

    turno è stato eliminato
  - [se blocco specificato e blocco.scadenza > *cal.*dataCorrente]

tab **non contiene** blocco

blocco è stato eliminato

Estensione 7a

Condizioni generali

- L'attore è identificato con un'istanza *org* di Organizzatore
- Esistenza di un'istanza *tab* di Tabellone Turni
- Esistenza di un'istanza *cal* di Calendario
- *cal* **coordina** *tab*
- *tab.*scadenzaGlobale > a *cal.*dataCorrente

7a. creaTurnoServizioRicorrente(turnoDiServizio: Turno di Servizio, cadenza: enumerazione, dataInizio: numero, dataFine: numero)

pre-condizioni:

- Che esista un'istanza *serv* di Service
- Che *tab* ***associato a*** *serv*
- *tab* **contiene** turnoDiServizio

post-condizioni:

- è stata creata un'istanza *ric* di Ricorrenza
- *cal* **coordina** *ric*
- turnoDiServizio **possiede** *ric*
- *ric*.cadenza = cadenza
- *ric.*dataInizio = dataInizio
- *ric.*dataFine = dataFine
- [se cadenza == settimanale]

  è stata creato un'elenco T di istanze *tusN* di Turno di Servizio con ogni istanza *tuS avente* attributo *tuS*.data = turnoDiServizio.data + 7 rispetto a *tuS -1, ognuna* con dataInizio < *tuS.data* < *dataFine*

- [se cadenza == mensile]

  è stata creato un'elenco T di istanze *tusN* di Turno di Servizio con ogni istanza *tuS avente* attributo *tuS*.data = turnoDiServizio.data + 30 rispetto a *tuS -1, ognuna* con dataInizio < *tuS.data* < *dataFine*

- [se cadenza == annuale]

  è stata creato un'elenco T di istanze *tusN* di Turno di Servizio con ogni istanza *tuS avente* attributo *tuS*.data = turnoDiServizio.data + 365 rispetto a *tuS -1, ognuna* con dataInizio < *tuS.data* < *dataFine*

- Ogni istanza *tuS* ***possiede*** *ric*

Estensione 9a-9b-9c-9d

Condizioni generali

- L'attore è identificato con un'istanza *org* di Organizzatore
- Esistenza di un'istanza *tab* di Tabellone Turni
- Esistenza di un'istanza *cal* di Calendario
- *cal* **coordina** *tab*
- *tab.*scadenzaGlobale > a *cal.*dataCorrente

9a. aggiungiScadenzaTurniTotali(data: Data)

pre-condizioni:

post-condizioni:

- L'attributo *tab.scadenzaGlobale* = data

9b. aggiungiScadenzaBlocco(dataScadenza:Data, blocco: Blocco)

pre-condizioni:

- *tab* **contiene** blocco

post-condizioni:

- blocco.scadenza = dataScadenza

9c. eliminaScadenzaTurno(scadenza: Data, turno: Turno)

pre-condizioni:

- l'attributo turno.scadenza > *cal.*dataCorrente

post-condizioni:

- dell'attributo turno.scadenza è stato eliminato

9d. eliminaScadenzaTurniTotali()

pre-condizioni:

- scadenza > *cal.*dataCorrente

post-condizioni:

- Il valore dell'attributo *tab.*scadenzaGlobale viene cancellato

Contratti Disponibilità

Scenario Principale

Precondizioni generali

- Esistenza di un'istanza *tab* di Tabellone Turni
- Esistenza di un'istanza *cal* di Calendario
- *cal* **coordina** *tab*
- *tab*.scadenzaGlobale > *cal.*dataCorrente

1. disponibilitàSingolo(turno: Turno, personale: Personale Di Servizio)

pre-condizione:

- *cal* **coordina** turno
- turno.scadenza > *cal.*dataCorrente
- turno.data > *cal.*dataCorrente

post-condizione:

- Istanza *perS* di Personale di Servizio **da disponibilità** a *t* istanza di Turno

2. eliminaDisponibilità(turno: Turno, personale: Personale Di Servizio)

pre-condizione

- *cal* **coordina** turno
- turno.scadenza > *cal.*dataCorrente
- personale **da disponibilità** a turno

post-condizione:

- è stata eliminata l'associazione personale **da disponibilità** a turno

Estensione 1a-1b

Precondizioni generali

- Esistenza di un'istanza *tab* di Tabellone Turni
- Esistenza di un'istanza *cal* di Calendario
- *cal* **coordina** *tab*
- *tab*.scadenzaGlobale > *cal.*dataCorrente

1a. disponibilitàBlocco(blocco: Turno, personale: Personale Di Servizio)

pre-condizione:

- Esistano almeno due istanze *t* di Turno Preparatorio ogni *t* **raggruppata in** blocco
- *cal* **coordina** blocco
- blocco.scadenza > *cal.*dataCorrente
- blocco.data > *cal.*dataCorrente

post-condizione:

- personale **da disponibilita** a blocco

1b. disponibilitàRicorrente(singoloTurnoOBloccoRic: Turno, ricorrenza: Ricorrenza, personale: Personale Di Servizio)

pre-condizione:

- singoloTurnoOBloccoRic **possiede** ricorrenza
- *cal* **coordina** singoloTurnoOBloccoRic
- singoloTurnoOBloccoRic.scadenza > *cal.*dataCorrente
- singoloTurnoRiccorrente.data > *cal.*dataCorrente

post-condizione:

- personale **da disponibilità a** singoloTurnoRicorrente

- Per ogni Turno *t,* in cui *t* **possiede** ricorrenza, personale **da disponibilita** a *t*

Estensione 2a-2b

Precondizioni generali

- Esistenza di un'istanza *tab* di Tabellone Turni
- Esistenza di un'istanza *cal* di Calendario
- *cal* **coordina** *tab*
- *tab*.scadenzaGlobale > *cal.*dataCorrente

2a. eliminaDisponibilitàBlocco(blocco: Turno, personale: Personale di Servizio)

pre-condizione

- *cal* **coordina** turno
- turno.scadenza > *cal.*dataCorrente
- personale **da disponibilità** a turno

post-condizione:

- è stata eliminata l'associazione personale **da disponibilità** a turno

2b. eliminaDisponibilitàRicorrente(singoloTurnoOBloccoRic: Turno, ricorrenza: Ricorrenza, personale : Personale di Servizio)

pre-condizione:

- singoloTurnoOBloccoRic **possiede** ricorrenza
- *cal* **coordina** singoloTurnoOBloccoRic
- singoloTurnoOBloccoRic.scadenza > *cal.*dataCorrente
- personale **da disponibilità** a singoloTurnoOBloccoRic
- Per ogni Turno *t,* in cui *t* **possiede** ricorrenza, personale **da disponibilità** a *t*

post-condizione:

- è stata eliminata l'associazione personale **da disponibilità** a singoloTurnoOBloccoRic
- Per ogni Turno *t,* in cui *t* **possiede** ricorrenza, viene eliminata l'associazione personale **da disponibilità** a *t*

Shift Management

ShiftManagementManager

event sender methods
+addReceiver(er: ShiftBoardReceiver)
+removeReceiver(er: ShiftBoardReceiver)

+notifyCreatedKitchenShift(shift: Shift)
+notifyCreatedServiceShift(shift: Shift)
+notifyModifiedShift(shift: Shift)
+notifyGlobalDeadlineSet(shiftBoard: ShiftBoard, date:Date)
+notifyLocalDeadlineSet(shift: Shift)
+notifyShiftDelete(shift: Shift)
+notifyAvailabilityGiven(shift: Shift, usr: User)
+notifyRecurrenceCreated(rec:Recurrence, shiftsOnDisplay: ArrayList<Shift>)
+notifyAvailabilityGiven (usr: User,shift: Shift)
+notifyRecurrentAvailabilityGiven(usr: User, shifts: ArrayList<Shifts>)
+notifyAvailabilityTaken(usr: User, shift: Shift)
+notifyRecurrentAvailabilityTaken(usr: User, shifts: ArrayList<Shifts>

operations methods
+checkEvent(ev: Event, serv: Service)
+getBoard(shiftBoard: ShiftBoard)
+setCurrentBoard(shiftBoard: ShiftBoard)
+setCurrentShift(shift: Shift)
+createBlock(shifts: ArrayList<KitchenShift>)
+modifyBlock(block: Block, shift: Shift)
+modifyShift(shift: Shift, beginTime?: String, endTime?: String, beforeHandTime?: Integer, aftermathTime?: Integer, location?: String)
+setLocalDeadline(shift: Shift, date: Date)
+createRecurrentBlocks(shift: Block, rec: Recurrence)
+createRecurrentBlocks(shift: Shift, rec: Recurrence)
+addShift(shift: Shift)
+createKitchenShift(date: Date, beginTime: Integer, endTime:Integer, location?: String)
+createServiceShift(date: Date, beginTime: Integer, endTime:Integer, afterMathTime?: Integer, beforehandTime?: Integer, location?: String)
+deleteShift(shift: Shift)
+checkShiftOverlap()
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

currentServiceBoard 0..1

ShiftBoard
-event: Event
-service: Service
-globalEndingDate: Date
-currentDate: Date
+createKitchenShift(date: Date, beginTime: Integer, endTime:Integer, location?: String)
+createServiceShift(date: Date, beginTime: Integer, endTime:Integer, afterMathTime?: Integer, beforehandTime?: Integer, location?: String)
+createBlock(shifts: ArrayList<KitchenShift>)
+setGlobalDeadline(date: Date)
+giveAvailability(shift: Shift, usr: User)
+checkShiftOverlap()
+tableSize()
+deleteShift(shift:Shift)
+getShift(shift: Shift)
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

shiftsOnDisplay 0..n

Shift
«Interface»
+setLocation;
+setAvailability;
+setLocalDeadline;
+isAvailable;
+getAvailabilitySize;
+isDeadlineSurpassed;
+createRecurrence;
+isRecurrent;
+deleteAvailability;

0..n currentShift 0..1
recurringShifts 0..n
recurrence 0..1

Recurrence
-startDate: Date
-endDate: Date
-frequence: {weekly, monthly, yearly}
+addToDisplayedShifts(shift: Shift)
+createShift(date: Date)
+deleteRecurrentAvailability(usr: User)
+giveRecurrentAvailability(usr: User)

ServiceShift
id: Integer
location; String
date: Date
beginTime: String
endTime: String
deadline: Date
aftermathTime: Integer
beforehandTime: Integer
availableStaffList: ArrayList<User>
recurrenceId: Integer
+setLocalDeadline(date: Date)
+setAvailability(usr: User)
+getAvailabilitySize()
+createRecurrence(rec: Recurrence)
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

KitchenShift
id: Integer
location; String
date: Date

beginTime: String
endTime: String

deadline: Date

availableStaffList: ArrayList<User>
recurrenceId: Integer
+setLocalDeadline(date: Date)
+setAvailability(usr: User)
+setRecurrenceId()
+getAvailabilitySize()
+isRecurrent(rec)
+createRecurrence(rec: Recurrence)
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

Block
id: Integer

date: Date

begintime: Integer
endTime: Integer

deadline: Date
availableStaffList: ArrayList<User>
+addShift(shift: KitchenShift)
+setLocalDeadline(date: Date)
+setAvailability(usr: User)
+getAvailabilitySize()
+createRecurrence(rec: Recurrence)
+deleteAvailability(shift: Shift, usr: User, recurrent: boolean)
+giveAvailability(shift: Shift, usr: User, recurrent: boolean)

eventReceivers 0..n

ShiftReceiver
{interface}
+*updateCreatedKitchenShift(shift:Shift)*
+*updateCreatedServiceShift(shift:Shift)*
+*updateModifiedShift(shift: Shift)*
+*updateLocalDeadlineSet(shift: Shift)*
+*updateShiftDeleted(shift: Shift)*
+*updateLocalDeadline(shift: Shift)*
+*updateModifiedBlock(block: Block)*
+*updateAvailabilityGiven(shift: Shift, usr: User)*
+*updateRecurrenceCreated(rec:Recurrence, shiftsOnDisplay: ArrayList<Shift>)*
+*updateRecurrentAvailabilityGiven(usr: User, shifts: ArrayList<Shifts>)*
+*updateAvailabilityGiven (usr: User,shift: Shift)*
+*notifyAvailabilityTaken(usr: User, shift: Shift)*
+*notifyRecurrentAvailabilityTaken(usr: User, shifts: ArrayList<Shifts>*

current Board

availabilities

shiftInUse 0..1

RecipeManagement

RecipeManagement

Event
Service
0..n
1 currentEvent
1 currentService
Event Manager

itemRecipe

Recipe
-name: String
0..n recipes
RecipeManager
+getRecipeBook(): ArrayList<Recipe>
+getPreparationBook(): ArrayList<Preparation>
Kitche Job
Preparation
- name: String
contains 0..n

jobList 0..n 1
job

Cucina Management

Summary Sheet
+addTask(kitchenJob: KitchenJob, coo?: User, sft: Shift)
+modifyTask(tsk: Task, kiS?: Shift, coo?: User)
+orderJobList(kitchenJob: KitchenJob, pos: int, originalPos: int)
+deleteTask(tsk: Task)
+markTaskReady(tsk: Task)
+addEstimateOrQuantity(tsk: Task, est? Integer, qt?: Integer)
+getKitchenJobPosition(job: KitchenJob)
+addKitchenJob(kitchenJob: KitchenJob)

itemTask 0..n

Task
id: Integer
quantity: Integer
estimate: Integer
completed: boolean
+markReady()
+addEstimateOrQuantity(est?: Integer, qt?: Integer)
+modifyTask(kitchenJob?:KitchenJob, shft?: Shift, coo?: User)

0..1 currentSheet

Summary Sheet Manager

event sender methods
+addReceiver(er: SummarySheetReceiver)
+removeReceiver(er: SummarySheetoReceiver)
-motifyTaskReady(sumSheet: SummarySheet, tsk: Task)
-notifyTaskCreated(sumSh: SummarySheet, tsk: Task)
-notifyTaskModified(sumSheet: SummarySheet, tsk: Task)
-notifyKitchenJobAdded(sumSh: SummarySheet,kitchenJob: KitchenJob)
-notifyTaskDeleted(sumSheet: SummarySheet, tsk: Task)
-notifyListRearranged(obj: Objective, pos: Integer)
-notifyEstimateOrQuantity(tsk: Task, est?: Integer, qt?: Integer)
-notifySummarySheetGenerated(sumSh: SummarySheet)

operations methods
+generateSummarySheet(ev: Event, serv: Service)
+openSummarySheet(sumSheet: SummarySheet)
+openShiftBoard(shiftBoard: ShiftBoard)
+getShiftBoard(kiSBoard: ShiftBoard)
+addTask(kitchenJob: Kitchen Job, kiS: Shift, coo? User)
+addKitchenJob(kitchenJob: KitchenJob)
+orderJobList(job: KitchenJob, posWanted: int, originalPos: int)
+markTaskReady(tsk: Task)
+addEstimateOrQuantityToTask(est? Integer, qt?: Integer)
+modifyTask(tsk?: Task, kitchenJob?: KitchenJob, kiS?: Shift, coo?: User)
+deleteTask(tsk: Task)
+getBoard(shiftBoard: ShiftBoard)
+setCurrentBoard(shiftBoard: ShiftBoard)
+setCurrentTask(tsk: Task)
+setCurrebtSheet(sumSh: SummarySheet)
+markTaskReady(tsk: Task)

SummarySheetReceiver
{interface}
+*updateSumSheetGenerated(sumSh: Summary Sheet)*
+*updateTaskCreated(tsk: Task, sumSheet: SummarySheet)*
+*updateModifiedTask(sumSheet: SummarySheet, tsk: Task)*
/+updateKitchenJobAdded(sumSh: SummarySheet, kitchenJob: KitchenJob)
+*updateTaskDeleted(sumSh: Summary Sheet, tsk: Task)*
+*updateListsRearrang(sumSheet: Summary Sheet, obj: Objective, pos: int,)*
/+updateTaskReady(sumShet: Summary Sheet, tsk: Task)/
+*updateEstimateOrQuantity(tsk: Task, est?: Integer, qt?: Integer)*
+*SummaryGenerated(sumSh: SummarySheet)*

0..n eventReceivers

cook

currentUser

owner

Design Class Diagram (DCD)

MenuManagement

MenuManager
-menuFeatures: String[]
event sender methods
+addReceiver(er: MenuEventReceiver)
+removeReceiver(er: MenuEventReceiver)
-notifySectionAdded(sec: Section)
-notifyMenuItemAdded(mi: MenuItem)
-notifyMenuCreated(m: Menu)
-notifyMenuDeleted(m: Menu)
-notifySectionsRearranged(m: Menu)
operations methods
+defineSection(name: String)
+inserItem(r: Recipe, sec?: Section, desc
+createMenu(title: String): Menu
+getRecipeBook( ): ArrayList<Recipe>

currentMenu 0..1

Menu
-title: String
-published: boolean
-inUse: boolean
-features: String[]
-featureValues: boolean[]
+create(owner: User, title: String, fea
+addSection(name: String)
+addItem(r: Recipe, sec?: Section,
+isOwner(user: User): boolean
+isInUse(): boolean
+hasSection(sec: Section): boolean
+sectionsSize(): int
+moveSection(sec: Section, pos: int)

sections 0..n

Section
name: String
+create(name: String)
+create(sec: Section)
+addItem(mi: MenuItem)

freeItems 0..n

MenuItem
description: String
+create(rec: Recipe, desc?: St
+create(mi: MenuItem)

sectionItems 0..n

MenuEventReceiver
{interface}
+*updateSectionAdded(m: Menu, sec: Section)*
+*updateMenuItemAdded(m: Menu, mi: MenuItem)*
+*updateMenuCreated(m: Menu)*
+*updateMenuDeleted(m: Menu)*
+*updateSectionsRearranged(m: Menu)*

General module

UseCaseLogicException
Exception

UserManagement

User
+isChef( ): boolean
+isCook(): boolean
+isPlanner(): boolean
+isWaitingStuff(): boolean
+giveAvailability(shift: Shift, recurrent: boolean)

1
0..1
0..n
0..1

0..1 currentUser

UserManager
event sender methods
operations methods
+getCurrentUser(): User

Detailed Sequence Diagram "Shifts"

[Shift] 1a createBlock

UI

shiftMngr:
Shift Manager

wantedShifts:
ArrayList<KitchenShifts>

shiftMngr.shiftBoard:
KitchenShiftBoard

shiftsOnDisplay:
ArrayList<Shift>

kiSBoardReceiver:
KitchenBoardReceiver

ALT usr.isManager() && shiftBoard != null && shiftBoard.currentDate.date< kiSBoard.globalEndingDate

createBlock(wantedShifts: ArrayList<KitchenShifts>)

createBlock(wantedShifts)

new Block(wantedShifts)

kitchenBlock:
Block

kitchenBlock.availableStaffList:
ArrayList<User>

create

LOOP for shift in wantedShifts

addShift(shift)

ALT if kitchenBlock.beginTime > shift.beginTime

this.beginTime = shift.beginTime

ALT if kitchenBlock.endTime < shift.endTime

this.endTime = shift.endTime

ALT if kitchenBlock.deadline > shift.deadline

this.deadline = shift.deadline

remove(shift)

kitchenBlock

add(kitchenBlock)

kitchenBlock

notifyBlockCreated(kitchenBlock)

LOOP for rec in Receivers

updateBlockCreated(kitchenBlock)

kitchenBlock

UseCaseLogic Exception

LOOP for eachShift in wantedShifts

ALT eachShift.date > shiftMngr.currentDate.date
&& eachShift.deadline > shiftMngr.currentDate
&& eachShift.availableCookList.size() == 0

UseCaseLogic Exception

UseCaseLogic Exception

[Shift] 1c createRecurrentBlock

UI

shiftMngr:
Shift Manager

shiftMngr.currentShift:
Block

shiftBoard.shiftsOnDisplay:
ShiftReceiver

shiftReceiver:
ShiftReceiver

ALT user.isOrganizer()

createRecurrentBlocks(shift: Block, rec: Recurrence)

setCurrentShift(shift)

createRecurrence(rec)

LOOP for shiftInBlock in currentShift.shiftsInside

ALT if !rec.equals(shiftInBlock.recurrence)

UseCaseLogic Exception

UseCaseLogic Exception

new Recurrence (block,rec)

recurrence
Recurrence

this.startDate = rec.startDate

this.endDate = rec.endDate

this.frequency = rec.frequency

create

recurrentShifts:
ArrayList<Shift>

add(block)

LOOP for
{i = startDate, i < endDate, i++}

createBlockForRec(beginTime, endTime, block.date, rec)

newBlock:
Block

create

kitchenBlock.availableStaffList:
ArrayList<User>

this.beginlTime = block.beginTime

this.endTime = block.endTime

ALT if rec.frequency == weekly

this.date =block.date+7

ALT if rec.frequency == monthly

this.date =block.date+30

ALT if rec.frequency == yearly

this.date =block.date+365

LOOP foreach(shift in block.shiftsInside

createShift(shift.date)

toAdd:
KitchenShift

ALT if rec.frequency == weekly

this.date =shift.date+7

ALT if rec.frequency == monthly

this.date =shift.date+30

ALT if rec.frequency == yearly

this.date =shift.date+365

add(toAdd)

add(newBlock)

addToDisplayedShifts(newBlock)

recurrence

recurrence

notifyRecurrenceAdded(recurrence, shiftsOnDisplay)

LOOP for sumSheeReceiver in Receivers

updateRecurrenceAdded(recurrence)

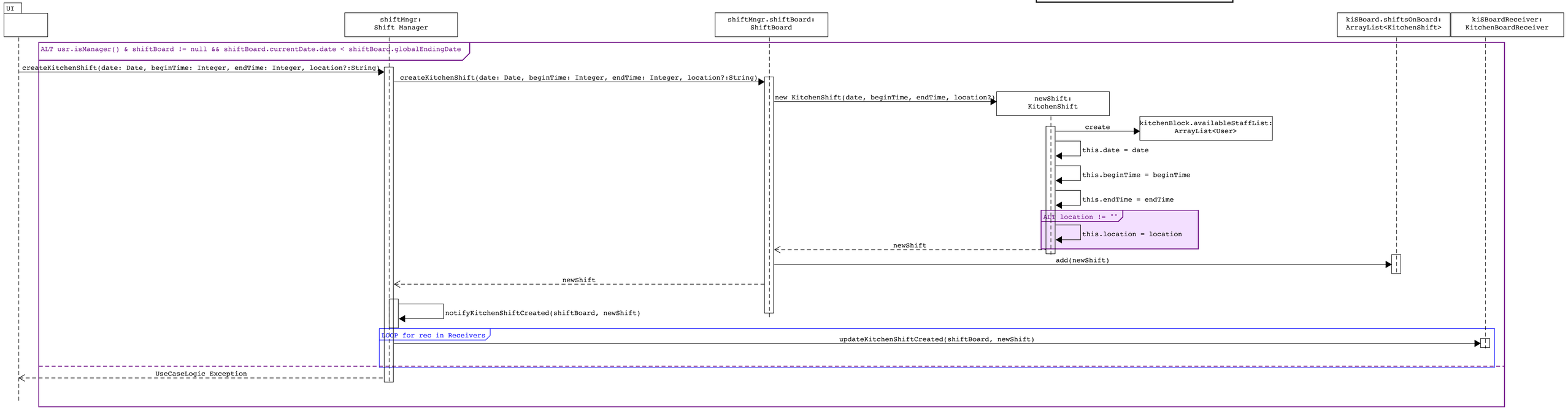
recurrence

UseCaseLogic Exception

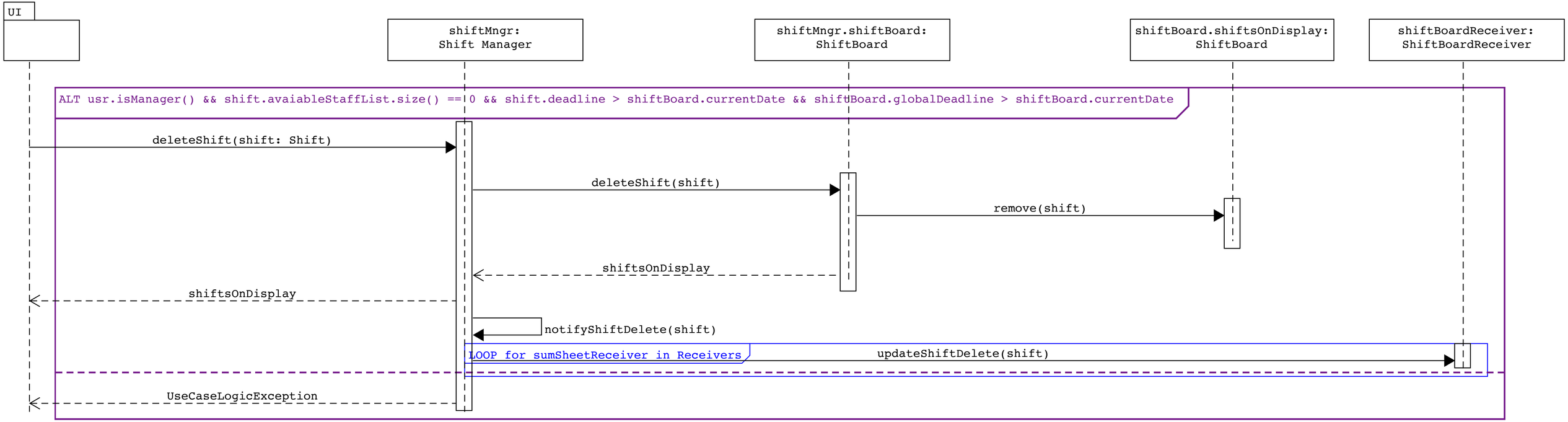
Note..
Create the shifts inside the block, check
if they're recursive with same frequence
if yes create them and add them to a block
and make the block recursive

```
recurrentShifts.add(block)

for( startDate, i < endDate, i++)
 Block newBlock = new Block()
 foreach(shift in block.shiftsInside)
  Shift toAdd = new Shift()
    if(rec.frequency == weekly)
     toAdd.originalDate + 7)
    if(rec.frequency == monthly)
     toAdd.originalDate + 30)
    if(rec.frequency == yearli)
     toAdd.originalDate + 365)
   newBlock.add(toAdd)

 newBlock.startTime = block.startTime
 newBlock.endTime = block.endTime

 if(rec.frequency == weekly)
    newBlock.date = bloc.date + 7
 if(rec.frequency == monthly)
     newBlock.date = bloc.date + 30)
 if(rec.frequency == yearly)
     newBlock.date = bloc.date + 365)

 recurrentShifts.add(newBlock)
 shiftsOnDisplay.add(newBlock)
```

[Shift] 2 createKitchenShift



[Shift] 5b deleteShift



[Shift] 9setLocalDeadline