Gestire i turni

Informazioni generali

Nome caso d'uso: Gestire i turni

Portata: Sistema

Livello: Obiettivo utente

Attore primario: Organizzatore

Parti Interessate: Organizzatore, Chef, Cuoco, Personale di servizio

Pre-condizioni:

• L'attore è autenticato come Organizzatore

Esistono eventi pianificati nel sistema per cui definire i turni

Garanzie di successo o post-condizioni:

- I turni (preparatori e di servizio) sono correttamente creati, modificati o eliminati nel sistema
- I raggruppamenti di turni preparatori sono definiti (se applicabile)
- I turni di servizio includono i tempi aggiuntivi specificati
- Le ricorrenze dei turni sono correttamente impostate (se applicabile)
- Il personale (cuochi e personale di servizio) può visualizzare i turni disponibili

Scenario principale di successo

#	Attore	Sistema
1	Consulta tabellone dei turni	Mostra tabellone dei turni esistente
2	Opzionalmente crea un nuovo turno di preparazione specificando inizio e fine del turno, data scadenza per la modifica ed eventualmente una sede.	Registra il turno sul tabellone dei turni
3	Opzionalmente crea un nuovo turno di servizio specificando inizio e fine del turno, data scadenza per la modifica, tempo di preparazione e di rigoverno.	Registra il turno sul tabellone dei turni
4	Opzionalmente imposta ricorrenza su turno presente.	Registra la ricorrenza di turni sul tabellone dei turni
5	Opzionalmente , seleziona almeno due turni preparatori esistenti e crea un raggruppamento.	Registra il raggruppamento sul tabellone dei turni
	Opzionalmente ripete dal passo 2 se no continua	
	Termina il caso d'uso	

Eccezione 2a

#	Attore	Sistema
2a.1	Opzionalmente crea un nuovo turno di preparazione specificando inizio e fine del turno, data scadenza per la modifica ed eventualmente una sede.	La sede indicata è satura
	Termina il caso d'uso	

Eccezione 2b

#	Attore	Sistema
2b.1	Opzionalmente crea un nuovo turno di preparazione specificando inizio e fine del turno, data scadenza per la modifica ed eventualmente una sede.	Le date inserite sono inconsistenti
	Termina il caso d'uso	

Estensione 2a

#	Attore	Sistema
2a.1	Modifica turno di preparazione specificando inizio e fine del turno, data scadenza per la modifica ed eventualmente una sede.	-

Eccezione (2a.1) a

#	Attore	Sistema
(2a.1) a.1	Modifica turno di preparazione specificando inizio e fine del turno, data scadenza per la modifica ed eventualmente una sede.	La sede indicata è satura per quell'orario
	Termina il caso d'uso	

Eccezione (2a.1) b

#	Attore	Sistema
(2a.1) b.1	Modifica turno di preparazione specificando inizio e fine del turno, data scadenza per la modifica ed eventualmente una sede.	Le date inserite sono inconsistenti
	Termina il caso d'uso	

Estensione 2b

#	Attore	Sistema
2b.1	Elimina turno di preparazione	Elimina il turno dal tabellone dei turni

Estensione 2c

#	Attore	Sistema
2c.1	Congela turno di preparazione per impedire il ritiro della disponibilità	Registra il turno modificato sul tabellone dei turni

Eccezione 3a

#	Attore	Sistema
3a.1	Opzionalmente crea un nuovo turno di servizio specificando inizio e fine del turno, data scadenza per la modifica, tempo di preparazione e di rigoverno.	
	Termina il caso d'uso	

Eccezione 3b

#	Attore	Sistema
3b.1	Opzionalmente crea un nuovo turno di servizio specificando inizio e fine del turno, data scadenza per la modifica, tempo di preparazione e di rigoverno.	Presenza di turni sovrapposti
	Termina il caso d'uso	

Estensione 3a

#	Attore	Sistema
3a.1	Modifica turno di servizio specificando inizio e fine del turno, data scadenza per la modifica, tempo di preparazione e di rigoverno.	

Eccezione (3a.1) a

#	Attore	Sistema
(3a.1) a.1	Modifica turno di servizio specificando inizio e fine del turno, data scadenza per la modifica, tempo di preparazione e di rigoverno.	Le date inserite sono inconsistenti
	Termina il caso d'uso	

Eccezione (3a.1) b

#	Attore	Sistema
(3a.1) b.1	Modifica turno di servizio specificando inizio e fine del turno, data scadenza per la modifica, tempo di preparazione e di rigoverno.	
	Termina il caso d'uso	

Estensione 3c

#	Attore	Sistema
3c.1	Congela turno di servizio per impedire il ritiro della disponibilità	Registra il turno modificato sul tabellone dei turni

Eccezione 4a

#	Attore	Sistema
4a.1	Opzionalmente imposta ricorrenza su turno presente.	Presenza di turni sovrapposti
	Termina il caso d'uso	

Estensione 4a

#	Attore	Sistema
4a.1	Elimina una ricorrenza di un turno	Elimina tutte le ricorrenze di un turno.

Estensione 4b

#	Attore	Sistema
4b.1	Modifica una ricorrenza di un turno	Modifica tutte le ricorrenze di un turno.

Eccezione (4b.1)a

#	Attore	Sistema
(4b.1)a.1	Modifica una ricorrenza di un turno	Presenza di turni sovrapposti
	Termina il caso d'uso	

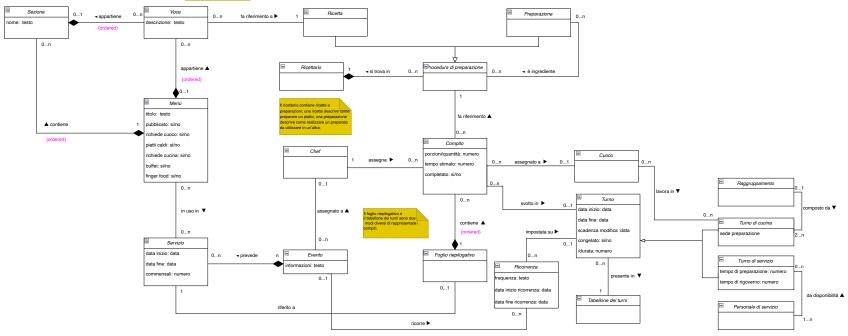
Eccezione (4b.1)b

#	Attore	Sistema
(4b.1)b.1	Modifica una ricorrenza di un turno	Le date inserite sono inconsistenti
	Termina il caso d'uso	

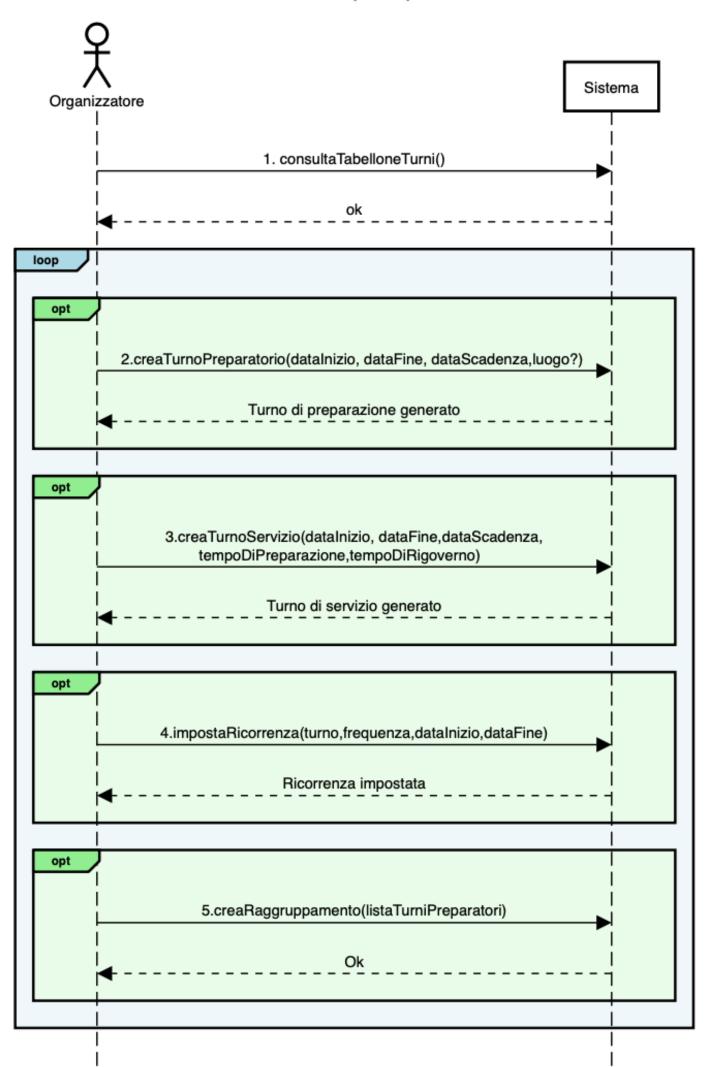
Estensione 5a

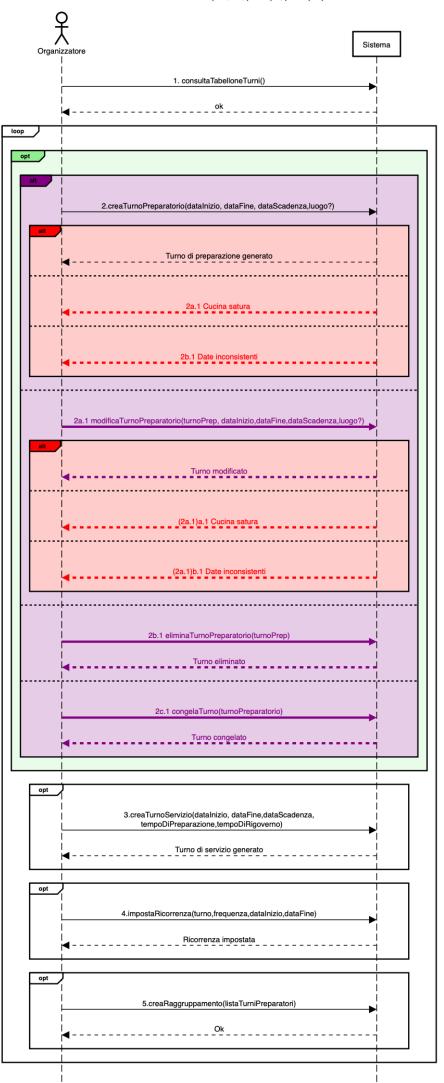
#	Attore	Sistema
5a.1	Scioglie un raggruppamento di turni	Elimina il raggruppamento mantenendo i singoli turni.



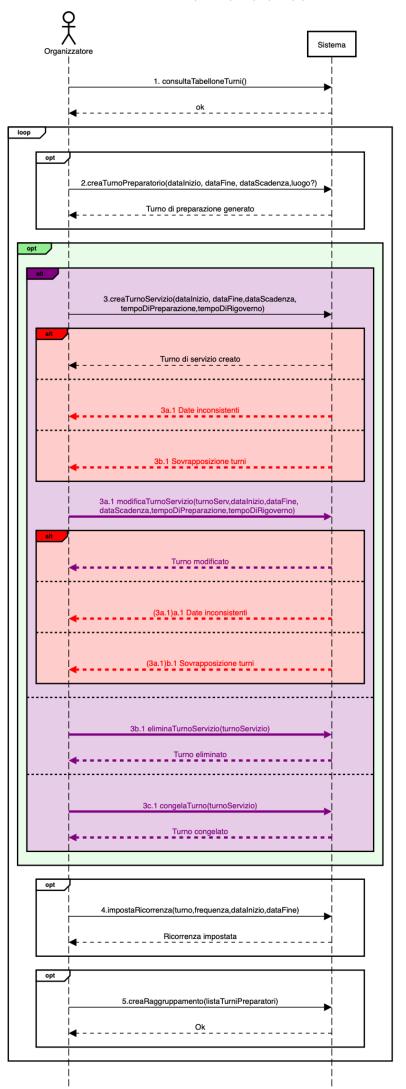


Scenario principale

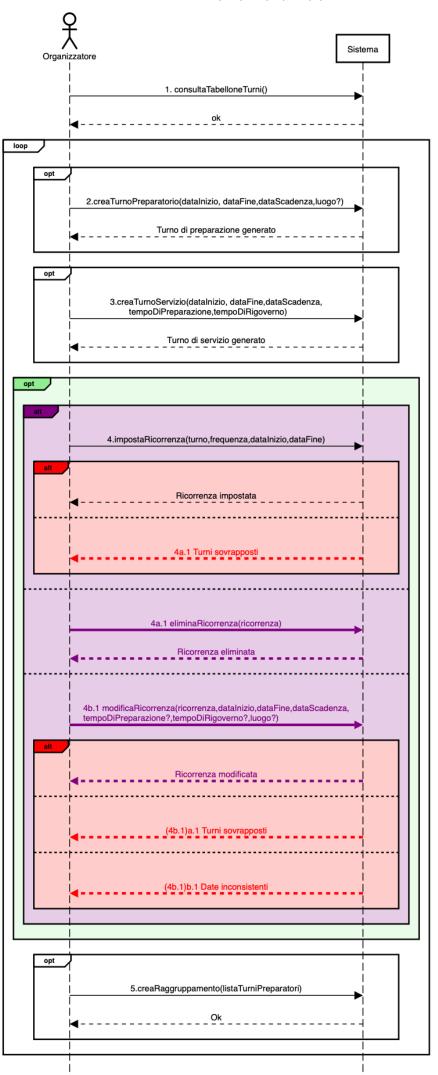




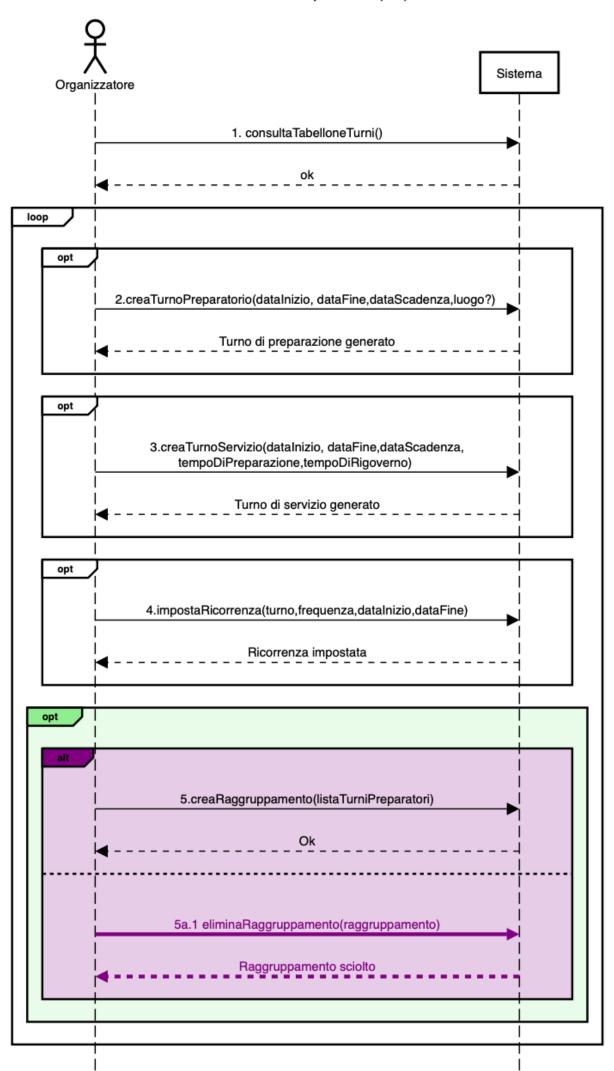
Estensioni al passo 3(3a,3b,3c) e relative eccezioni (3a,3b,(3a.1)a,(3a.1)b)



Estensioni al passo 4(4a,4b) e relative eccezioni (4a,(4b.1)a,(4b.1)b)



Estensioni al passo 5(5a)



Contratti per lo UC "Gestire i turni"

Pre-condizione generale: l'attore è identificato con un'istanza *org* di Organizzatore, a *org* è assegnato <u>evento</u> a cui appartiene <u>servizio</u>

1.consultaTabelloneTurni()

Precondizioni:

- esiste un'istanza tabellone di Tabellone turni

Post-condizioni: ---

2.creaTurnoPreparatorio(<u>dataInizio</u>:data, <u>dataFine</u>:data, <u>dataScadenza</u>:data, <u>luogo?</u>:SedeCucina)

Pre-condizioni:

Esiste un'istanza cucinaPred di SedeCucina

Post-condizioni:

Se <u>datalnizio</u>, <u>dataFine</u> e <u>dataScadenza</u> non sono date passate, <u>datalnizio</u> è antecedente a <u>dataFine</u> e <u>dataScadenza</u> è compresa tra <u>datalnizio</u> e <u>dataScadenza</u>:

Se <u>luogo</u> è specificato e non esiste in *tabellone* un'istanza *turno_i* di TurnoPreparatorio svolto in <u>luogo</u> tale che l'intervallo tra *turno_i.datalnizio* e *turno_i.dataFine* si sovrapponga all'intervallo <u>datalnizio</u> e <u>dataFine</u>:

- è stata creata un'istanza turnoPreparatorio di Turno Preparatorio
- turnoPreparatorio è presente in tabellone
- turnoPreparatorio.datalnizio = datalnizio
- turnoPreparatorio.dataFine = dataFine
- turnoPreparatorio.dataScadenza = dataScadenza
- turnoPreparatorio.congelato = no
- turnoPreparatorio è svolto in <u>luogo</u>

Se <u>luogo</u> non è specificato e non esiste in *tabellone* un'istanza *turno_i* di TurnoPreparatorio svolto in *cucinaPred* tale che l'intervallo tra *turno_i.dataInizio* e *turno_i.dataFine* si sovrapponga all'intervallo <u>dataInizio</u> e <u>dataFine</u>:

- è stata creata un'istanza turnoPreparatorio di Turno Preparatorio
- turnoPreparatorio è presente in tabellone
- turnoPreparatorio.datalnizio = datalnizio
- turnoPreparatorio.dataFine = dataFine
- turnoPreparatorio.dataScadenza = dataScadenza
- turnoPreparatorio.congelato = no
- turnoPreparatorio è svolto in cucinaPred

2a.1 modificaTurnoDiPreparazione(<u>turnoPreparatorio</u>:TurnoDiPreparazione, <u>dataInizio</u>:data, <u>dataFine</u>:data, <u>dataScadenza</u>:data, <u>luogo</u>?:SedeCucina)

Pre-condizioni:

- <u>turnoPreparatorio</u> è presente in *tabellone*
- esiste un'istanza di SedeCucina associata a turnoPreparatorio.

Post-condizioni:

Se <u>datalnizio</u>, <u>dataFine</u> e <u>dataScadenza</u> non sono date passate, <u>datalnizio</u> è antecedente a <u>dataFine</u> <u>dataScadenza</u> è compresa tra <u>datalnizio</u> e <u>dataScadenza</u> e <u>turnoPreparatorio.congelato</u> = no:

Se <u>luogo</u> è specificato e non esiste in *tabellone* un'istanza *turno_i* di TurnoPreparatorio svolto in <u>luogo</u> tale che l'intervallo tra *turno_i.datalnizio* e *turno_i.dataFine* si sovrapponga all'intervallo datalnizio e dataFine:

- è stata modificata l'istanza <u>turnoPreparatorio</u>
- turnoPreparatorio è presente in tabellone
- turnoPreparatorio.datalnizio = datalnizio
- turnoPreparatorio.dataFine = dataFine
- turnoPreparatorio.dataScadenza = dataScadenza
- turnoPreparatorio è svolto in <u>luogo</u>

Se <u>luogo</u> non è specificato e non esiste in *tabellone* un'istanza *turno_i* di TurnoPreparatorio, associata alla stessa istanza SedeCucina di <u>turnoPreparatorio</u>, tale che l'intervallo tra *turno_i.datalnizio* e *turno_i.dataFine* si sovrapponga all'intervallo <u>datalnizio</u> e <u>dataFine</u>:

- è stata modificata l'istanza <u>turnoPreparatorio</u>
- turnoPreparatorio è presente in tabellone
- turnoPreparatorio.datalnizio = datalnizio
- turnoPreparatorio.dataFine = dataFine
- turnoPreparatorio.dataScadenza = dataScadenza

Se <u>turnoPreparatorio</u> è contenuto in un'istanza *ric* di Ricorrenza questa viene modificata.

2b.1 eliminaTurnoDiPreparazione(turnoPreparatorio:TurnoDiPreparazione)

Pre-condizioni:

- <u>turnoPreparatorio</u> è presente in *tabellone*

Post-condizioni:

- -l'istanza <u>turnoPreparatorio</u> viene eliminata
- se turnoPreparatorio era contenuta in un'istanza ric di Ricorrenza questa viene eliminata

2c.1 congelaTurnoPreparatorio(turnoPreparatorio:TurnoDiPreparazione)

Pre-condizioni:

- <u>turnoPreparatorio</u> è presente in *tabellone* e *turnoPreparatorio.congelamento* = no

Post-condizioni:

- Il turnoPreparatorio viene congelato e non è più possibile ritirare una disponibilità
- turnoPreparatorio.congelamento = si
- Se <u>turnoPreparatorio</u> è contenuto in un'istanza *ric* di Ricorrenza guesta viene congelato.

3.creaTurnoServizio(<u>dataInizio</u>:data, <u>dataFine</u>:data, <u>dataScadenza</u>:data, <u>tempoDiPreparazione</u>:numero,<u>tempoDiRigoverno</u>:numero)

Pre-condizioni: ---

Post-condizioni:

Se <u>datalnizio</u>, <u>dataFine</u> e <u>dataScadenza</u> non sono date passate, <u>datalnizio</u> è antecedente a <u>dataFine</u> e <u>dataScadenza</u> è compresa tra <u>datalnizio</u> e <u>dataScadenza</u>:

Se non esiste in *tabellone* un'istanza *turno_i* di TurnoDiServizio tale che l'intervallo tra *turno_i.datalnizio* e *turno_i.dataFine* si sovrapponga all'intervallo <u>datalnizio</u> e <u>dataFine</u>:

- è stata creata un'istanza turnoDiServizio di TurnoDiServizio
- turnoDiServizio è presente in tabellone
- turnoDiServizio.datalnizio = datalnizio
- turnoDiServizio.dataFine = dataFine
- turnoDiServizio.dataScadenza = dataScadenza
- turnoDiServizio.tempoDiPreparazione = tempoDiPreparazione
- *turnoDiServizio.tempoDiRigoverno* = <u>tempoDiRigoverno</u>
- turnoPreparatorio.congelato = no

3a.1 modificaTurnoServizio(<u>turnoDiServizio</u>:TurnoDiServizio, <u>dataInizio</u>:data, <u>dataFine</u>:data, <u>dataScadenza</u>:data, <u>tempoDiPreparazione</u>:numero, <u>tempoDiRigoverno</u>:numero)

Pre-condizioni:

- <u>turnoDiServizio</u> è presente in *tabellone*

Post-condizioni:

Se <u>datalnizio</u>, <u>dataFine</u> e <u>dataScadenza</u> non sono date passate, <u>datalnizio</u> è antecedente a <u>dataFine</u> e <u>dataScadenza</u> è compresa tra <u>datalnizio</u> e <u>dataScadenza</u>:

Se non esiste in *tabellone* un'istanza *turno_i* di TurnoDiServizio tale che l'intervallo tra *turno_i.datalnizio* e *turno_i.dataFine* si sovrapponga all'intervallo <u>datalnizio</u> e <u>dataFine</u>:

- è stata modificata l'istanza turnoDiServizio
- <u>turnoDiServizio</u> è presente in *tabellone*
- turnoDiServizio.datalnizio = datalnizio
- turnoDiServizio.dataFine = dataFine
- turnoDiServizio.dataScadenza = dataScadenza
- turnoDiServizio.tempoDiPreparazione = tempoDiPreparazione
- *turnoDiServizio.tempoDiRigoverno* = <u>tempoDiRigoverno</u>
- turnoPreparatorio.congelato = no

Se <u>turnoDiServizio</u> è contenuto in un'istanza *ric* di Ricorrenza questa viene modificata.

3b.1 eliminaTurnoServizio (turnoServizio: TurnoDiServizio)

Pre-condizioni:

- turnoServizio è presente in tabellone

Post-condizioni:

- -l'istanza turnoServizio viene eliminata
- se turnoServizio era contenuta in un'istanza ric di Ricorrenza questa viene eliminata

3c.1 congelaTurnoDiServizio(<u>turnoServizio</u>: TurnoDiServizio)

Pre-condizioni:

- turnoServizio è presente in tabellone e turnoServizio.congelato = no

Post-condizioni:

- Il turnoServizio viene congelato e e non è più possibile ritirare una disponibilità
- turnoServizio.congelato= si

Se <u>turnoDiServizio</u> è contenuto in un'istanza *ric* di Ricorrenza questa viene congelata.

4.impostaRicorrenza(<u>turno</u>:Turno,<u>frequenza</u>:testo,<u>dataInizio</u>:data,<u>dataFine</u>:data)

Pre-condizioni:

<u>turno</u> è presente in *tabellone*, <u>datalnizio</u> e <u>dataFine</u> non sono date passate, <u>datalnizio</u> è antecedente a <u>dataFine</u> *e* <u>frequenza</u> è una frequenza valida (giornaliera, settimanale, mensile).

Post-condizioni:

Se non esiste un Turno t tale che si sovrapponga ad una ricorrenza di turno:

- È stata creata una Ricorrenza *r* associata a <u>turno</u>
- *r.frequenza* = <u>frequenza</u>
- *r.datalnizio* = datalnizio
- r.dataFine = dataFine

- Per ogni ricorrenza di turno da datalnizio a dataFine crea un'istanza turno_i dove:
 - turno i.datalnizio = turno.datalnizio
 - turno_i.dataFine = turno.dataFine
 - turno i.dataScadenza = turno.dataScadenza
 - -[se <u>turno</u> è un Turno di preparazione] *turno_i. luogoPreparazione = turno. luogoPreparazione*
 - -[se <u>turno</u> è un Turno di servizio] *turno_i. tempoDiPreparazione* = *turno. tempoDiPreparazione*
 - -[se <u>turno</u> è un Turno di servizio] *turno_i. tempoDiRigoverno = turno. tempoDiRigoverno*

4.a eliminaRicorrenza(<u>ric</u>:Ricorrenza)

Pre-condizioni:

Post-condizioni:

Ogni *turno_i* che ricorre in <u>ric</u> viene eliminato L'istanza ric viene eliminata

4.b modificaRicorrenza(<u>ric</u>:Ricorrenza, <u>dataInizio</u>:data, <u>dataFine</u>:data, <u>dataScadenza</u>:data,<u>tempoDiPreparazione?</u>:numero, <u>tempoDiRigoverno?</u>:numero, <u>luogo</u>?:SedeCucina)

Pre-condizioni:

Se si vogliono modificare dei turni di servizio occorre definire <u>tempoDiPreparazione</u> e <u>tempoDiRigoverno</u> e lasciare non definito <u>luogo</u>. In alternativa se si vogliono modificare dei turni di preparazione occorre non definire <u>tempoDiPreparazione</u> e <u>tempoDiRigoverno</u> mentre <u>luogo</u> è opzionale.

Post-condizioni:

Se <u>datalnizio</u>, <u>dataFine</u> e <u>dataScadenza</u> non sono date passate, <u>datalnizio</u> è antecedente a <u>dataFine</u> e <u>dataScadenza</u> è compresa tra <u>datalnizio</u> e <u>dataScadenza</u> e

Se non esiste un Turno t tale che si sovrapponga ad un $turno_i$ che ricorre in ric dopo la modifica:

Ogni *turno_i* che ricorre in <u>ric</u> viene modificato:

- turno_i.dataInizio = turno.dataInizio + ric.frequenza * numero della ricorrenza
- turno_i.dataFine = turno.dataFine + ric.frequenza * numero della ricorrenza
- turno_i.dataScadenza = turno.dataScadenza
- -[se <u>turno</u> è un Turno di preparazione] *turno_i. luogoPreparazione* = *turno. luogoPreparazione*
- -[se <u>turno</u> è un Turno di servizio] *turno_i. tempoDiPreparazione = turno. tempoDiPreparazione*
- -[se <u>turno</u> è un Turno di servizio] *turno_i. tempoDiRigoverno* = *turno. tempoDiRigoverno*

5. creaRaggruppamento(<u>listaTurniPreparatori</u>: lista TurnoDiPreparazione)

Precondizioni:

- Tutti i turni in <u>listaTurniPreparatori</u> sono istanze di TurnoDiPreparazione
- <u>listaTurniPreparatori</u> contiene almeno 2 istanze di TurnoDiPreparazione

Post-condizioni:

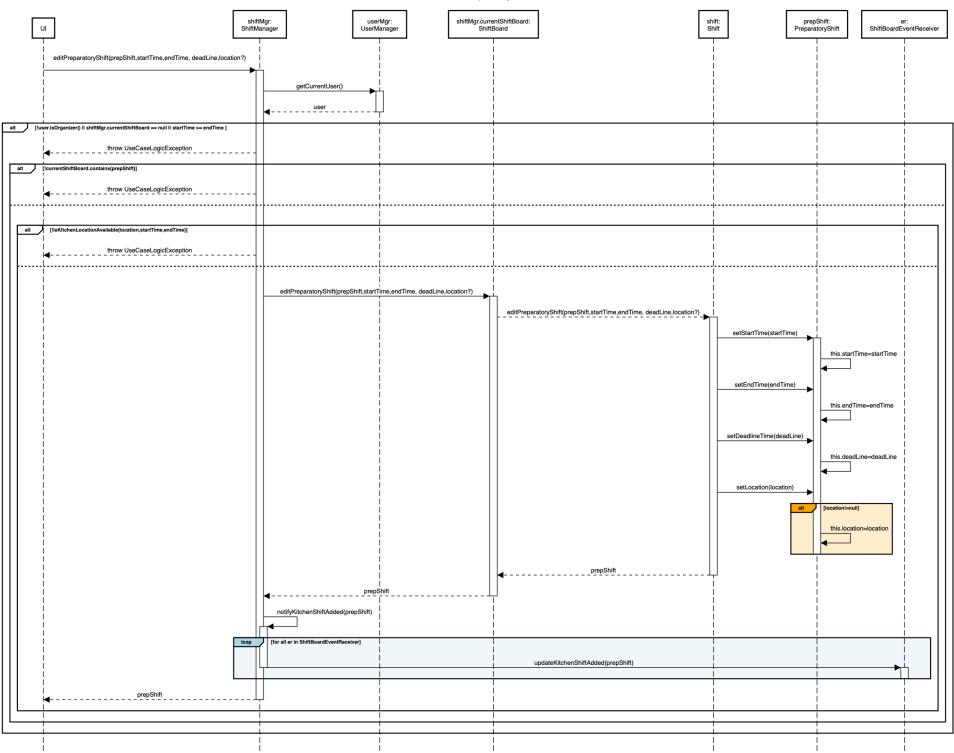
- È stato creato un nuovo Raggruppamento r
- Tutti i turni in <u>listaTurniPreparatori</u> sono contenuti in *r*

5a.1 sciogliRaggruppamento(raggruppamento: Raggruppamento)

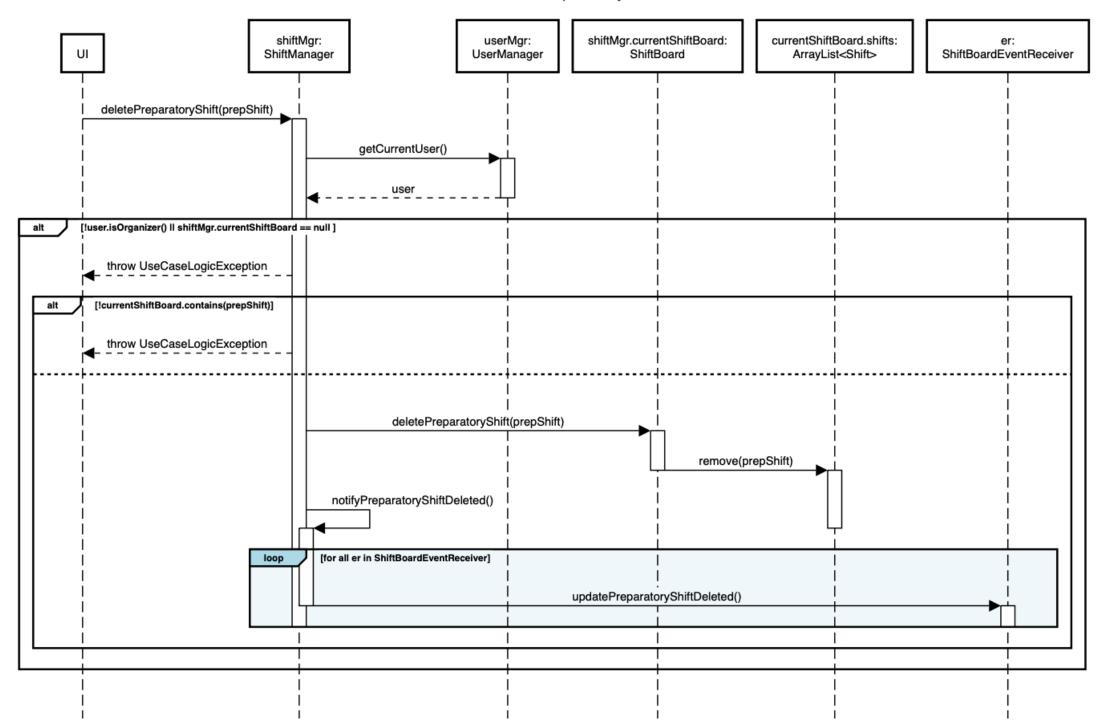
Precondizioni:

Post-condizioni:

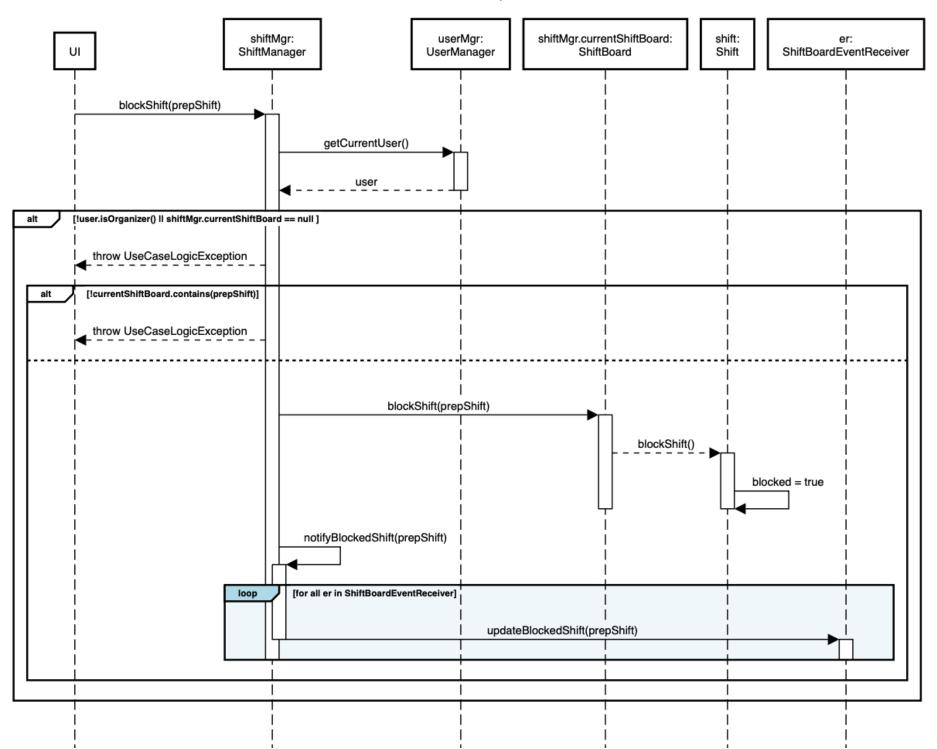
- Per ogni TurnoDiPreparazione t che è contenuto in <u>raggruppamento</u>, l'associazione tra <u>raggruppamento</u> a t è eliminata.
- <u>raggruppamento</u> è eliminata



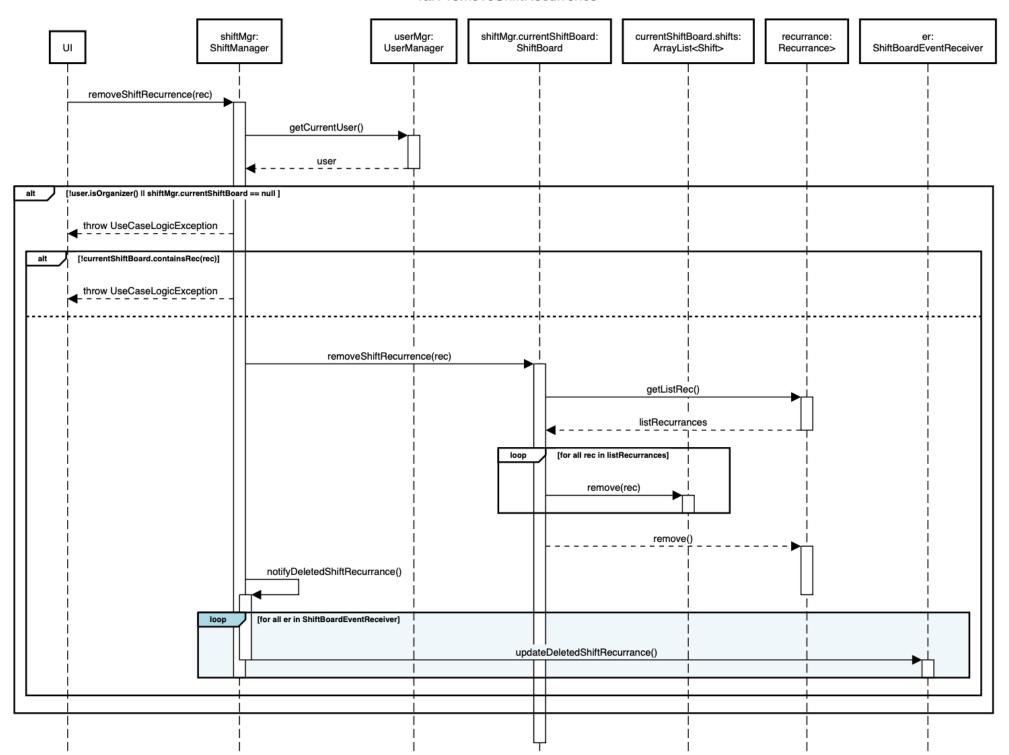
2b.1 deletePreparatoryShift



2c.1 blockPrepShift



4a.1 removeShiftRecurrence



Design Class Diagram (DCD) MenuManagement General module UserManagement MenuManager UseCaseLogicException Exception Menu -menuFeatures: String[] -title: String event sender methods -published: boolean UserManager +addReceiver(er: MenuEventReceiver) -inUse: boolean +removeReceiver(er: MenuEventReceiver) -features: String[] -notifySectionAdded(sec: Section) event sender methods -featureValues: boolean[] User -notifyMenuItemAdded(mi: MenuItem) +create(owner: User, title: String, features: String) -notifvMenuCreated(m: Menu) +addSection(name: String) currentUser -notifyMenuDeleted(m: Menu) +isChef(): boolean +addItem(r: Recipe, sec?: Section, operations methods currentMenu -notifySectionsRearranged(m: Menu) +isOrganizer(): boolean +getCurrentUser(): User +isCook(): boolean +isOwner(user: User): boolean +isServiceStaff(): boolean operations methods +isInUse(): boolean +defineSection(name: String) +hasSection(sec: Section): boolean +inserItem(r: Recipe, sec?: Section, desc?: String) +sectionsSize(): int +createMenu(title: String): Menu | sections | +moveSection(sec: Section, pos: int) +getRecipeBook(): ArrayList<Recipe> eventReceivers Section ServiceStuff Chef Cook Organizer name: String MenuEventReceiver +create(name: String) {interface} +isUserAvailable(shift:Shift) +create(sec: Section) : boolean +addItem(mi: MenuItem) Menultem +updateSectionAdded(m: Menu. sec: Section) +updateMenuItemAdded(m: Menu, mi: MenuItem) description: String +updateMenuCreated(m: Menu) -create(rec: Recipe, desc?: String) rupdateMenuDeleted(m: Menu) sectionItems +create(mi: Menultem) menu +updateSectionsRearranged(m: Menu) refers SummarySheet Management EventManagement SheetManager SummarySheet EventManager Event Service -startDate: Date -description: String -id: int -startDate: Date -endDate: Date event sender methods getCurrentEvent() rrentEvent -endDate: Date getEventOwner(event: Event): Ch -notifySheetCreated(sheet: SummarySheet) + addTask(procedure: Procedure) services -numPerson: int -notifySheetDeleted(sheet: SummarySheet) + removeTask(task: Task) -info: String -notifyTaskCreated(task: Task) + sortTask(task: Task, position:int) +getOwner(): Chef notifyTaskDeleted() + modifyTask(task:Task,requiredTime?: int, -notifyTaskModified(task: Task) quantity?: int, portions?: int) -notifyTaskSorted(sheet: SummarySheet) + setTaskCompleted(task: Task) + hasTask(task: Task) : boolean +addReceiver(sr: SheetEventReceiver) +removeReceiver(sr: SheetEventReceiver) + tasksSize(): int operations methods +createSummarySheet(service:Service, event:Event): SummarySheet +setCurrentSheet(sheet: SummarySheet) +openSummarySheet():SummarySheet +deleteSheet(sheet: SummarySheet) +createTask(procedure: Procedure): Task +deleteTask(task: Task) +sortTask(task:Task, position: int) +assignTask(task:Task, cook: Cook, shift:PreparatoryShift, requiredTime: int, quantity: int, portions: int) +modifyTask(task:Task, cook: Cook shift:PreparatoryShift, requiredTime: int, Task quantity: int, portions: int) -quantity: int +setTaskPrepared(task: Task) -portion: int +showShiftBoard() -requiredTime: Time + hasTask(task: Task) : boolean RecipeBookManagement -completed: boolean + setCompleted() + setQuantity(quantity) refers + setPortion(portion) + setRequiredTime(requiredTime) eventReceivers KitchenProcedure RecipeBook SheetEventReceiver -name: String {interface} -itemBooks: List < ItemBook> +updateSheetCreated(sheet: SummarySheet) book +updateSheetDeleted(sheet: SummarySheet) +updateItemAdded(sheet: SummarySheet, item:ItemBook) +updateAssignmentCreated(sheet: SummarySheet,asg:Assignment) +updateAssignmentPortion(asg:Assignment, portion: int) +updateAssignmentTime(sheet: SummarySheet, time: Time) +updateSummarySheetNotes(note: String) +updateCookInAssignment(asg:Assignment, cook: Cook) +updateTurnInAssignment(asg:Assignment, turn: Turn) +updateItemModified(task: Recipe, asg: Assignment) +updateAssignmentDeleted(asg: Assignment) cook +updateTaskSorted(sheet: SummarySheet) Assignment setTask(task: Task) getTask(): Task RecipeBookManager setPreparatoryShift(prepShift: PreparatoryShift) getPreparatoryShift(): PreparatoryShift Recipe Preparation setCook(cook: Cook) event sender methods getCook(): Cook -name: String -name: String preparations operations methods +getRecipes(): ArrayList<ItemBook>

TurnManagement ShiftManager ShiftBoard event sender methods + addReceiver(sr: ShiftEventReceiver) + createPreparatoryShift(startTime: DateTime,endTime:DateTime, currentShiftBoard + removeReceiver(sr: ShiftEventReceiver) deadLine: DateTime, location?: kitchenLocation): Preparatory S notifyCreatedShift(s: Shift) 1 + createServiceShift(startTime: DateTime, endTime:DateTime.deadLine:DateTime prepTime: Time, rigTime: Time): ServiceShift notifyDeletedShift() notifyModifiedShift(s: Shift) + editPreparatoryShift(prepShift: PreparatoryShift,startTime: DateTime, notifyBlockedShift() endTime:DateTime,deadLine: DateTime,location?: kitchenLocation? notifyCreatedShiftRecurrance(recShift: ShiftRecurrence) + deletePreparatoryShift(prepShift: PreparatoryShift) - notifyModifiedShiftRecurrance(recShift: ShiftRecurrence) + blockShift(shift: Shift) notifyDeletedShiftRecurrance() + editServiceShift(servShift:ServiceShift,startTime:DateTime, endTime:DateTime, deadLine:DateTime, prepTime: Time, rigTime: Time) - notifyCreatedShiftGroup(group :ShiftGroup) notifyDeletedShiftGroup() + deleteServiceShift(servShift:ServiceShift) + createShiftRecurrence(shift: Shift,fregUnit: TimeUnit, startTime: DateTime, endTime: DateTime):ShiftRecurrence operations methods + removeShiftRecurrence(recShift: ShiftRecurrence) + getCurrentShiftBoard(): ShiftBoard recurrances + editShiftRecurrence(recShift: ShiftRecurrence, startTime:DateTime, + setCurrentShiftBoard(shiftBrd: ShiftBoard) endTime:DateTime.deadLine:DateTime, prepTime?: Time. + createPreparatoryShift(startTime: DateTime.endTime:DateTime. rigTime?: Time, location?: kitchenLocation) deadLine: DateTime, location?: kitchenLocation): Preparator + createShiftGroup(listPrepShift:List < PreparatoryShift>): ShiftGroup reateServiceShift(startTime: DateTime, endTime:DateTime,deadLine:DateTime + deleteShiftGroup(group :ShiftGroup) prepTime: Time, rigTime: Time): ServiceShift + containsShift(shift: Shift): boolean + editPreparatoryShift(prepShift: PreparatoryShift,startTime: DateTime, + getShift(shift:Shift): Shift endTime:DateTime.deadLine: DateTime.location?: kitchenLd + getListShift() : List <Shift> + deletePreparatoryShift(prepShift: PreparatoryShift) ShiftGroup + blockShift(shift: Shift) + editServiceShift(servShift:ServiceShift,startTime:DateTime, endTime:DateTime, deadLine:DateTime, prepTime: Time, rigTime: Time) + addShift(shift: PreparatoryShift):PreparatoryShift + deleteServiceShift(servShift:ServiceShift) + removeShift(shift: PreparatoryShift) createShiftRecurrence(shift: Shift,freqUnit: TimeUnit, + getShiftGroup(): List < PreparatoryShift> startTime: DateTime, endTime: DateTime):ShiftRecurre + removeShiftRecurrence(recShift: ShiftRecurrence) Shift + editShiftRecurrence(recShift: ShiftRecurrence, startTime:DateTime endTime:DateTime,deadLine:DateTime, prepTime?: Time, - id: int rigTime?: Time, location?: kitchenLocation) - blocked : boolean + createShiftGroup(listPrepShift:List < PreparatoryShift>): ShiftGroup + deleteShiftGroup(group :ShiftGroup) + createPreparatoryShift(startTime: DateTime,endTime:DateTime, + isKitchenLocationAvailable(location: KitchenLocation, startTime: DateTime, deadLine: DateTime, location?: kitchenLocation): Preparatory endTime: DateTime): boolean + createServiceShift(startTime: DateTime, endTime:DateTime,deadLine:DateTime prepTime: Time, rigTime: Time): ServiceShift + editPreparatoryShift(prepShift: PreparatoryShift,startTime: DateTime, endTime:DateTime.deadLine: DateTime.location?: kitchenLod + deletePreparatoryShift(prepShift: PreparatoryShift) + editServiceShift(servShift:ServiceShift,startTime:DateTime, endTime:DateTime. deadLine:DateTime, prepTime: Time, rigTime: Time) - deleteServiceShift(servShift:ServiceShift) + createShiftRecurrence(fregUnit: TimeUnit,startTime: DateTime, endTime: DateTime):ShiftRecurrence + removeShiftRecurrence(recShift: ShiftRecurrence) + editShiftRecurrence(recShift: ShiftRecurrence, startTime:DateTime, endTime:DateTime,deadLine:DateTime, prepTime?: Time, ShiftEventReceiver Recurrance rigTime?: Time, location?: kitchenLocation) + blockShift() {interface} -freq: String, |shifts |+ getBlocked() : boolean -startTime: DateTime, + getRecurrance() : Reccurrance -endTime: DateTime) updateCreatedShift(s: Shift) + getDuration(): int - updateDeletedShift() updateModifiedShift(s: Shift) + setFreq(freq) updateBlockedShift() + setStartTime(startTime) - updateCreatedShiftRecurrance(recShift: ShiftRecurrence + setEndTime(endTime) updateModifiedShiftRecurrance(recShift: ShiftRecurrend + getListRec(): List <Shift> - updateDeletedShiftRecurrance() + removeShift(shift:Shift) - updateCreatedShiftGroup(group :ShiftGroup) + editShift(shift:Shift) recurrances - updateDeletedShiftGroup() KitchenLocation PreparatoryShift ServiceShift - startTime: DateTime - startTime: DateTime -name : String - endTime: DateTime endTime: DateTime sets - deadline: DateTime - deadline: DateTime - prepTime: Time - rigTime: Time + setStartTime(startTime: DateTime) + setEndTime(endTime:DateTime) + setStartTime(startTime: DateTime) + setDeadLine(deadLine: DateTime) + setEndTime(endTime:DateTime) + setLocation(location: kitchenLocation) + setDeadLine(deadLine: DateTime) + getStartTime():DateTime + setPrepTime(prepTime: Time) | + setRigTime(rigTime: Time) + getEndTime():DateTim + getDeadLine():DateTim + getLocation(): KitchenLocation + getStartTime():DateTime + getEndTime():DateTime + getDeadLine():DateTime + getPrepTime(): Time + getRigTime(): Time

work