

# Gestire gli eventi

Bettarello, Mendoza - Progetto SAS Laboratorio 22/23

# Informazioni generali

**Nome caso d'uso:** Gestire gli eventi

**Portata:** Sistema

**Livello:** Obiettivo Utente

**Attore primario:** Organizzatore

**Parti Interessate:** Chef, personale di servizio, cuoco

**Pre-condizioni:** L'attore deve essere autenticato come Organizzatore

**Garanzie di successo o post-condizioni:** L'evento è terminato

## Scenario principale di successo

#	Attore	Sistema
1	Compila la scheda per l'evento	Creata la scheda per l'evento
2	Assegna uno Chef alla cucina	Chef assegnato all'evento
3	Effettua modifiche al menù	Modifiche al menù effettuate
<i>Se desidera torna al passo 3 se no prosegue</i>		
4	Assegna il personale ai turni	Personale di servizio assegnato ai turni
<i>Se parte del personale non e' disponibile ritorna al passo 4 se no prosegue</i>		
5	<b>Opzionalmente</b> , assegna un cuoco ad un determinato servizio	Cuoco assegnato per le preparazioni dell'ultimo minuto
6	<b>Opzionalmente</b> , definisce la ricorrenza dell'evento	Ricorrenza dell'evento definita
7	<b>Opzionalmente</b> , si aggiungono eventuali note e documentazione al termine dell'evento	Aggiunte la documentazione e le note riguardanti l'evento svoltosi
<i>Termine del caso d'uso</i>		

### Estensione 1a

#	Attore	Sistema
1a.1	Apri una scheda evento pre-esistente	Aperta per eventuale modifica la scheda di un evento pre-esistente

### Eccezione 1a.1a

#	Attore	Sistema
1a.1a.1	Apri una scheda evento pre-esistente	L'evento è in corso e la scheda non può essere aperta da nessuno
Termine del caso d'uso		

### Estensione 1b

#	Attore	Sistema
1b.1	Elimina un evento esistente	Eliminato un evento esistente
Termine del caso d'uso		

### Eccezione 1b.1a

#	Attore	Sistema
1b.1a.1	Elimina un evento esistente	L'evento è in corso e non può essere eliminato
Termine del caso d'uso		

### Estensione 1c

#	Attore	Sistema
1c.1	Annulla un evento esistente	Annullato un evento esistente

<i>Termine del caso d'uso</i>
-------------------------------

Eccezione 1c.1a

#	Attore	Sistema
1c.1a.1	Elimina un evento esistente	L'evento è in corso e non può essere annullato
<i>Termine del caso d'uso</i>		

Estensione 6a

#	Attore	Sistema
6a.1	Modifica la ricorrenza di un evento pre-esistente	Aperta per modifica la scheda di un evento ricorrente pre-esistente
<i>Termine del caso d'uso</i>		

Eccezione 6a.1a

#	Attore	Sistema
6a.1a.1	Modifica la ricorrenza di un evento pre-esistente	La ricorrenza è in corso non può essere modificata
<i>Termine del caso d'uso</i>		

Estensione 6b

#	Attore	Sistema
6b.1	Elimina una ricorrenza di un evento esistente	Eliminata la ricorrenza di un evento esistente

## Eccezione 6b.1a

#	Attore	Sistema
6b.1a.1	Elimina una ricorrenza di un evento esistente	La ricorrenza è in corso e non può essere eliminata
<i>Termine del caso d'uso</i>		

## Modello di dominio

mdd here



## Diagramma di sequenza di sistema

# Modello di dominio

# Contratti

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

**1. creaEvento( titolo : testo, inizio: testo, fine: testo, luogo: testo, elegante: booleano, privato: booleano, ospiti: numero, servizi: Servizio..\* )**

**pre-condizioni:**

- *L'Organizzatore or è stato assegnato a un evento*

**post-condizioni:**

- è stata creata un'istanza *e* di Evento
- *e.titolo* = titolo
- *e.inizio* = inizio
- *e.fine* = fine
- *e.luogo* = luogo
- *e.elegante* = elegante
- *e.privato* = privato
- *e.ospiti* = ospiti
- *or* è proprietario di *e*
- Per ogni elemento serv di servizi viene creata un'istanza *s* di Servizio ed viene associata ad *e*

**2. assegnaChef ( chef : Chef )**

**pre-condizioni:**

- è in corso la definizione di una evento *e*
- non esiste un *e'* evento in cui *e'.inizio* = *e.inizio* dove chef è assegnato a *e'*

**post-condizioni:**

- chef è assegnato ad *e*

### **3. propostaModificheMenu ( menu: Menu, modifica: Modifica...\* )**

#### **pre-condizioni:**

- esiste un'istanza *m* di menu tale che menu = *m*;

#### **post-condizioni:**

- è creata un'istanza della modifica chiamata *mod*
- L'istanza *mod* è associata a *m*

### **4. assegnaPersonale ( personale : Personale...\*, turnoServizio: Turno)**

#### **pre-condizioni:**

- è in corso la definizione di un evento *e*
- esiste un'istanza *sse* di Turno di servizio dove turnoServizio = *sse*

#### **post-condizioni:**

- personale è assegnato a *sse*

### **5. assegnaCuocoAServizio ( cuoco : Personale, preparatorio: Turn)**

#### **pre-condizioni:**

- è in corso la definizione di una Evento *e*
- esiste un'istanza *sp* di Turno preparatorio dove preparatorio= *sp*

#### **post-condizioni:**

- cuoco è assegnato a preparatorio

## **6. definisceRicorrenza ( event: Evento, data: testo, contFrequenza: numero )**

### **pre-condizioni:**

- L'Organizzatore *or* è stato assegnato a un evento

### **post-condizioni:**

- è stata creata un'istanza *r* di Ricorrenza
- *r.event* = event
- *r.data* = data
- *r.contFrequenza* = contFrequenza

## **7. aggiuntaDocumentazione ( file: file)**

### **pre-condizioni:**

- è in corso la definizione di una evento *e*
- se l'orario attuale - *e.inizio* è maggiore di *s.fineOffset*

### **post-condizioni:**

- è stata creata un'istanza *d* di Documentazione
- *d.file* = file

# Contratti delle estensioni

## 1a. apreSchedaEvento ( titolo: testo)

### pre-condizioni:

- *or* deve essere proprietario di evento
- la scheda evento non è in uso

## 1b. eliminaEvento ( titolo: testo)

### pre-condizioni:

- *or* deve essere proprietario di evento
- la scheda evento non è in uso

### post-condizioni:

- Se *or* è proprietario di evento e evento non è in uso:
- evento è eliminata.

## 1c annullaEvento (evento : evento)

### pre-condizioni:

- *or* deve essere proprietario di evento
- la scheda evento non è in uso

### post-condizioni:

- evento è annullato.

## 6a. modificaRicorrenza ( event?: Evento, data? : data, contFrequenza? : numero )

### pre-condizioni:

- è in corso la definizione di una ricorrenza *r*

### post-condizioni:

- [ se event è specificato ]: *r.event* = event
- [ se data è specificato ]: *r.data* = data
- [ se contFrequenza è specificato ]: *r.contFrequenza* = contFrequenza

**6b. eliminaRicorrenza (event: Evento, data: testo, contFrequenza: numero )**

**pre-condizioni:**

- è in corso la definizione di una ricorrenza *r*

**post-condizioni:**

- *r.evento*, *r.data*, *r.contFrequenza* sono eliminati dall'oggetto *r*

# Design Class Diagram



```

classDiagram
    package EventManagement {
        class Variation {
            -item: MenuItem
            -isAccepted: boolean
            +create(item: MenuItem, type: Variation)
            +isAccepted(): boolean
            +setItem(item: MenuItem)
            +setType(type: VariationType)
            +setAccepted(isAccepted: boolean)
        }
        class VariationType {
            enum DELETE, EDIT, ADD
        }
        class MenuEventReceiver {
            <<interface>>
            +updateMenuCreated(m: Menu)
            +updateMenuDeleted(m: Menu)
            +updateVariationCreated(v: Variation)
            +updateSectionAdded(m: Menu, sec: Section)
            +updateMenuItemAdded(m: Menu, mi: MenuItem)
            +updateMenuFeaturesChanged(m: Menu)
            +updateMenuTitleChanged(m: Menu)
            +updateMenuPublishedState(m: Menu)
            +updateSectionDeleted(m: Menu, s: Section)
            +updateSectionChangedName(m: Menu, s: Section)
            +updateSectionsRearranged(m: Menu)
            +updateFreeItemsRearranged(m: Menu)
            +updateItemSectionChanged(m: Menu, s: Section, mi: MenuItem)
            +updateItemDescriptionChanged(m: Menu, mi: MenuItem)
            +updateItemDeleted(m: Menu, sec: Section, mi: MenuItem)
        }
        class EventEventReceiver {
            <<interface>>
            +updateEventCreated(e: Event)
            +updateEventDeleted(e: Event)
            +updateEventModified(e: Event)
            +updateRecurrencyCreated(r: Recurrency)
            +updateRecurrencyDeleted(r: Recurrency)
            +updateRecurrencyModified(r: Recurrency)
            +updateServiceCreated(serv: Service)
            +updateServiceDeleted(serv: Service)
            +updateServiceModified(serv: Service)
        }
        class EventManager {
            -currentEvent: Event
            <<event sender methods>>
            <<operations methods>>
        }
        class Penal {
            -description: string
            -sum: float
            +create(sum: float, description: String)
            +setSum(sum: float)
            +setDescription(description: String)
            +getSum(): float
            +getDescription(): String
        }
        class Section {
            -name: string
            +create(s: Section)
            +addItem(mi: MenuItem)
            +updateItems(newItems: ArrayList<MenuItem>)
            +getItemPosition(mi: MenuItem): int
            +testString(): String
            +toString(): String
            +setName(name: String)
            +moveItem(mi: MenuItem, position: int)
            +removeItem(mi: MenuItem)
        }
        class RecurrencyType {
            enum DAILY, WEEKLY, MONTHLY, YEARLY
        }
        class Recurrency {
            -event: Event
            -date: Date
            -recurrencyCount: int
            +create(event: Event, date: Date, recurrencyCount: int)
            +setRecurrencyCount(recurrencyCount: int)
            +setEvent(event: Event)
            +setDate(date: Date)
            +getEvent(): Event
            +getDate(): Date
            +getRecurrencyCount(): int
            +getRecurrenciesOfEvent(): ArrayList<Recurrency>
        }
        class MenuItem {
            -description: string
            -itemRecipe: Recipe
            +create(rec: Recipe)
            +create(rec: Recipe, desc: String)
            +create(mi: MenuItem)
            +toString(): String
            +setDescription(description: String)
            +setItemRecipe(itemRecipe: Recipe)
        }
    }

    package UserManagement {
        class UserManager {
            -currentUser: User
            <<event sender methods>>
            <<operations methods>>
        }
        class User {
            -username: string
            +create()
            +isChef(): boolean
            +isCook(): boolean
            +isOrganizer(): boolean
            +getUserRole(): UserRole
            +toString(): String
        }
        class UserRole {
            enum MANAGER, COOK, CHEF, WAITER
        }
        class Shift {
            -start: Date
            -end: Date
            -limit: int
            -attendances: ArrayList<User>
            -shiftMap: HashMap<Integer, Shift>
            +create(start: string, end: string, type: ShiftType, limit: int)
            +setStart(start: Date)
            +setEnd(end: Date)
            +setType(type: ShiftType)
            +setLimit(limit: int)
            +getAttendances(): ArrayList<Shift>
            +isInTimeframe(start: Date, end: Date)
            +toString(): String
        }
        class ShiftManager {
            <<event sender methods>>
            <<operations methods>>
        }
        class ShiftType {
            enum PREPARATORY, SERVICE
        }
        class ShiftEventReceiver {
            <<interface>>
            +updateShiftCreated(sh: Shift)
            +updateShiftDeleted(sh: Shift)
            +updateShiftModified(sh: Shift)
        }
    }

    package KitchenManagement {
        class TaskListOrder {
            enum ByImportance, byDifficulty
        }
        class Task {
            -name: string
            -ingredients: string
            -staffInstructions: string
            -notes: string
            -recipe: Recipe
            -preparation: Preparation
            -start: Date
            -end: Date
            -asseegee: User
            -importance: int
            -difficulty: int
            -order: int
            +setName(name: String)
            +setIngredients(ingredients: String)
            +setStaffInstructions(staffInstructions: String)
            +setNotes(notes: String)
        }
        class Recipe {
            -name: string
            -timeToPrepare: Time
            +create()
            +create(name: String): Recipe
            +getTimeToPrepare(): Time
            +getName(): String
            +getDifficulty(): int
            +toString(): String
        }
        class RecipeManager {
            <<event sender methods>>
            <<operation methods>>
        }
        class Preparation {
            -name: string
            -description: string
            +create(name: String): Preparation
            +getName(): String
            +setName(name: String)
            +getDescription(): String
            +setDescription(description: String)
        }
        class KitchenEventReceiver {
            <<interface>>
            +updateSummarySheetCreated(sh: SummarySheet)
            +updateSummarySheetDeleted(sh: SummarySheet)
            +updateTaskCreated(t: Task, s: SummarySheet)
            +updateTaskDeleted(t: Task)
            +updateTaskModified(t: Task, s: SummarySheet)
            +updateRecipeCreated(re: Recipe)
            +updateRecipeDeleted(re: Recipe)
            +updateRecipeModified(re: Recipe)
            +updatePreparationCreated(p: Preparation)
            +updatePreparationDeleted(p: Preparation)
            +updatePreparationModified(p: Preparation)
            +updateSummarySheetDeleted(s: SummarySheet)
        }
        class Kitchen {
            <<event sender methods>>
            <<operation methods>>
        }
        class KitchenManager {
            -currentKitchen: Kitchen
            <<event sender methods>>
            <<operation methods>>
        }
        class SummarySheet {
            -order: TaskListOrder
            -editable: boolean
            -service: Service
            -owners: ArrayList<User>
            -taskList: ArrayList<Task>
            +create(): SummarySheet
            +create(order: boolean, service: Service): SummarySheet
            +getSummarySheetsTaskList(s: SummarySheet): ArrayList<Task>
            +getSummarySheetOfTask(t: Task): SummarySheet
            +setOwners(owner: ArrayList<User>)
            +getOwners(): ArrayList<User>
            +setOrder(order: TaskListOrder)
            +orderTaskList(order: TaskListOrder)
            +setEditable(flag: boolean)
            +addTaskToSheet(t: Task)
            +getTaskList(): ArrayList<Task>
            +setTaskList()
            +setService(s: Service)
            +move(index: int, t: Task)
            +toString(): String
        }
    }

    EventManager --> Variation : proposes
    EventManager --> Event : creates
    EventManager --> Section : has
    EventManager --> MenuItem : has
    EventManager --> Recurrency : has
    EventManager --> Penal : has
    EventManager --> EventEventReceiver : eventReceivers
    EventManager --> MenuEventReceiver : menuReceivers
    EventManager --> ShiftManager : shiftReceivers
    EventManager --> KitchenManager : kitchenReceivers
    EventManager --> SummarySheet : summaryReceivers

    UserManager --> User : currentUser
    UserManager --> Shift : works
    UserManager --> ShiftManager : currentShift
    UserManager --> ShiftEventReceiver : shiftReceivers
    UserManager --> KitchenManager : currentKitchen
    UserManager --> SummarySheet : summaryReceivers

    KitchenManager --> TaskListOrder : compiles
    KitchenManager --> Task : contains
    KitchenManager --> Recipe : produces
    KitchenManager --> Preparation : composes
    KitchenManager --> Kitchen : currentKitchen
    KitchenManager --> KitchenEventReceiver : kitchenReceivers
    KitchenManager --> SummarySheet : summaryReceivers

    Variation --> MenuManager : 0..n currentVariation
    Variation --> MenuItem : 0..1
    MenuManager --> Menu : 0..1 currentMenu
    MenuManager --> MenuItem : 0..n menuReceivers
    MenuManager --> Section : 0..n
    MenuManager --> MenuItem : 0..1
    MenuManager --> Event : 0..n
    MenuManager --> Recurrency : 0..n
    MenuManager --> Penal : 0..n
    MenuManager --> EventEventReceiver : 0..n
    MenuManager --> MenuEventReceiver : 0..n
    MenuManager --> ShiftManager : 0..1
    MenuManager --> ShiftEventReceiver : 0..n
    MenuManager --> KitchenManager : 0..n
    MenuManager --> SummarySheet : 0..1

    User --> Shift : 1..n works
    User --> ShiftManager : 1..1 organizes
    User --> KitchenManager : 0..n
    User --> SummarySheet : 0..1
    User --> Event : 0..n
    User --> Recurrency : 0..n
    User --> Penal : 0..n
    User --> EventEventReceiver : 0..n
    User --> MenuEventReceiver : 0..n
    User --> ShiftManager : 0..1
    User --> ShiftEventReceiver : 0..n
    User --> KitchenManager : 0..n
    User --> SummarySheet : 0..1

    Shift --> ShiftManager : 0..1 currentShift
    Shift --> KitchenManager : 0..n
    Shift --> SummarySheet : 0..1
    Shift --> Event : 0..n
    Shift --> Recurrency : 0..n
    Shift --> Penal : 0..n
    Shift --> EventEventReceiver : 0..n
    Shift --> MenuEventReceiver : 0..n
    Shift --> ShiftManager : 0..1
    Shift --> ShiftEventReceiver : 0..n
    Shift --> KitchenManager : 0..n
    Shift --> SummarySheet : 0..1

    ShiftManager --> Shift : 0..1 currentShift
    ShiftManager --> KitchenManager : 0..n
    ShiftManager --> SummarySheet : 0..1
    ShiftManager --> Event : 0..n
    ShiftManager --> Recurrency : 0..n
    ShiftManager --> Penal : 0..n
    ShiftManager --> EventEventReceiver : 0..n
    ShiftManager --> MenuEventReceiver : 0..n
    ShiftManager --> ShiftManager : 0..1
    ShiftManager --> ShiftEventReceiver : 0..n
    ShiftManager --> KitchenManager : 0..n
    ShiftManager --> SummarySheet : 0..1

    KitchenManager --> TaskListOrder : 0..n
    KitchenManager --> Task : 0..n
    KitchenManager --> Recipe : 0..n
    KitchenManager --> Preparation : 0..n
    KitchenManager --> Kitchen : 0..1
    KitchenManager --> KitchenEventReceiver : 0..n
    KitchenManager --> SummarySheet : 0..n
    KitchenManager --> Event : 0..n
    KitchenManager --> Recurrency : 0..n
    KitchenManager --> Penal : 0..n
    KitchenManager --> EventEventReceiver : 0..n
    KitchenManager --> MenuEventReceiver : 0..n
    KitchenManager --> ShiftManager : 0..1
    KitchenManager --> ShiftEventReceiver : 0..n
    KitchenManager --> KitchenManager : 0..n
    KitchenManager --> SummarySheet : 0..1

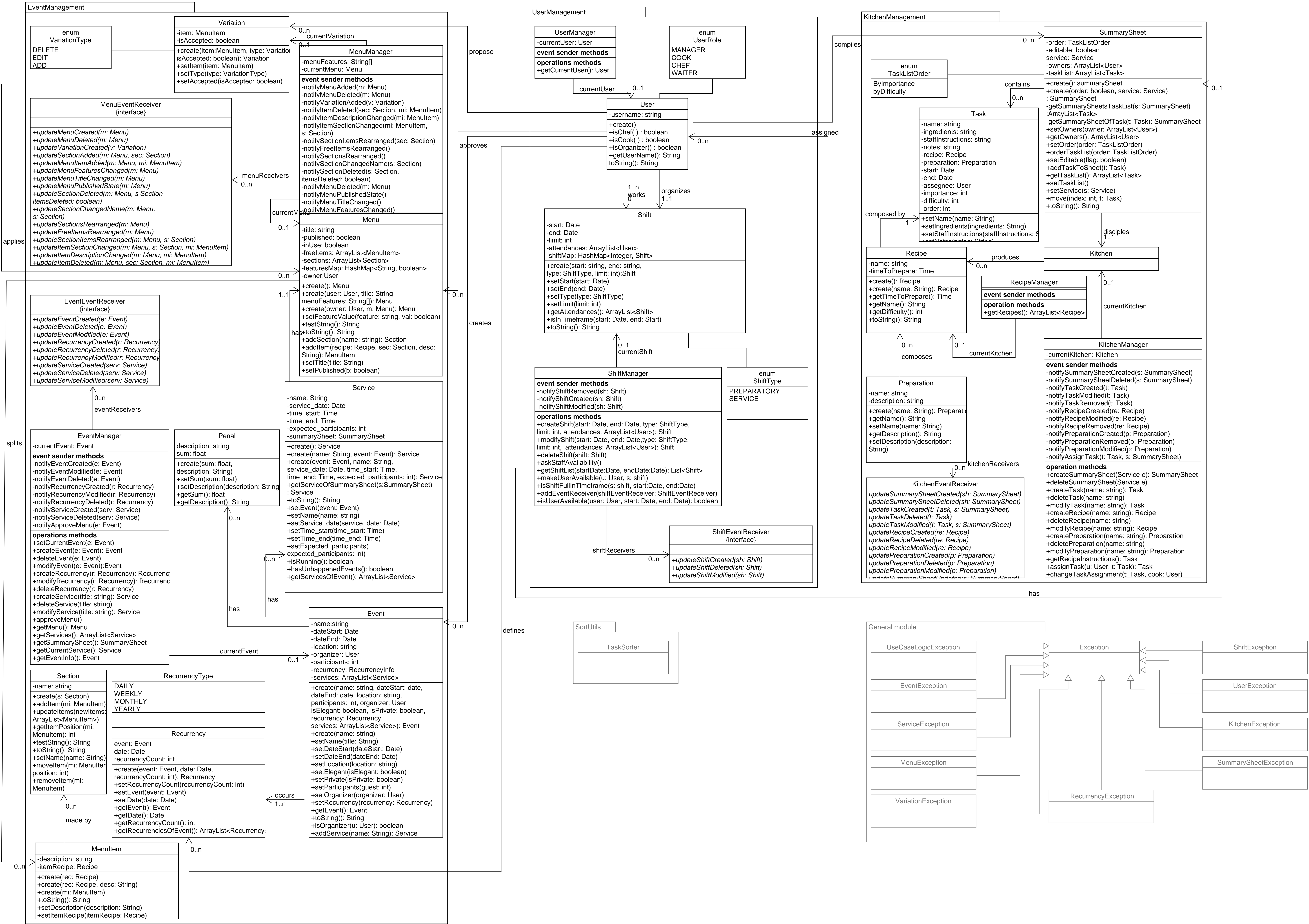
    SummarySheet --> TaskListOrder : 0..n
    SummarySheet --> Task : 0..n
    SummarySheet --> Recipe : 0..n
    SummarySheet --> Preparation : 0..n
    SummarySheet --> Kitchen : 1..1
    SummarySheet --> KitchenEventReceiver : 0..n
    SummarySheet --> SummarySheet : 0..n
    SummarySheet --> Event : 0..n
    SummarySheet --> Recurrency : 0..n
    SummarySheet --> Penal : 0..n
    SummarySheet --> EventEventReceiver : 0..n
    SummarySheet --> MenuEventReceiver : 0..n
    SummarySheet --> ShiftManager : 0..1
    SummarySheet --> ShiftEventReceiver : 0..n
    SummarySheet --> KitchenManager : 0..n
    SummarySheet --> SummarySheet : 0..1

    Event --> Section : 0..n
    Event --> MenuItem : 0..n
    Event --> Recurrency : 0..n
    Event --> Penal : 0..n
    Event --> EventEventReceiver : 0..n
    Event --> MenuEventReceiver : 0..n
    Event --> ShiftManager : 0..1
    Event --> ShiftEventReceiver : 0..n
    Event --> KitchenManager : 0..n
    Event --> SummarySheet : 0..1

    Recurrency --> Section : 0..n
    Recurrency --> MenuItem : 0..n
    Recurrency --> Recurrency : 0..n
    Recurrency --> Penal : 0..n
    Recurrency --> EventEventReceiver : 0..n
    Recurrency --> MenuEventReceiver : 0..n
    Recurrency --> ShiftManager : 0..1
    Recurrency --> ShiftEventReceiver : 0..n
    Recurrency --> KitchenManager : 0..n
    Recurrency --> SummarySheet : 0..1

    Penal --> Section : 0..n
    Penal --> MenuItem : 0..n
    Penal --> Recurrency : 0..n
    Penal --> Penal : 0..n
    Penal --> EventEventReceiver : 0..n
    Penal --> MenuEventReceiver : 0..n
    Penal --> ShiftManager : 0..1
    Penal --> ShiftEventReceiver : 0..n
    Penal --> KitchenManager : 0..n
    Penal --> SummarySheet : 0..1

    MenuItem
```



# System Sequence Diagram

CreaEvento

PropostaModificheMenu

AssegnaCuocoAServizio

DefinisceRicorrenza

ModificaRicorrenza

AggiuntaDocumentazione





