Sommario

2. Elaborazione – Iterazione 2

2.1 Introduzione

La fase di elaborazione ha l'obiettivo di: raffinare la Visione, implementare il nucleo dell'architettura del software, risolvere le eventuali problematiche collegate ai rischi, individuare i requisiti e fornire stime realistiche del piano di lavoro. In questa fase si analizzano e si implementano i seguenti casi d'uso:
UC2: Gestione Ordine, questo caso d'uso permette ai clienti di effettuare ordini di piatti e bevande dal menu.

UC3: Gestione Pagamento, questo caso è gestito dall'amministratore e si occupa della gestione dei pagamenti per ogni tavolo.

UC5: Modifica ordine già effettuato, permette a un cliente di apportare modifiche a un ordine già inviato, purché non sia ancora in preparazione. Il cliente può aggiungere, modificare o eliminare portate.

UC8: Fidelity Card, questo caso d'uso gestisce il programma fedeltà per i clienti.

2.2 Analisi Orientata agli Oggetti

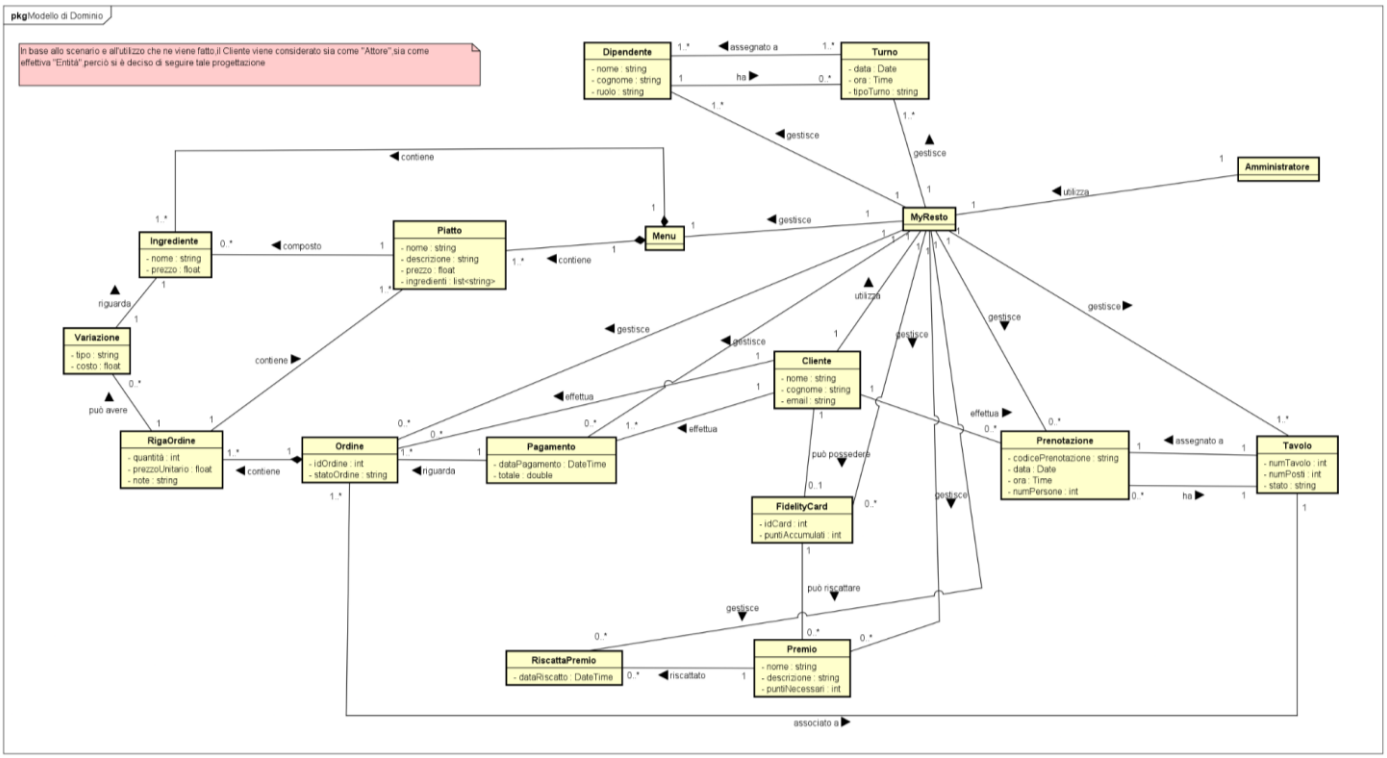
L'analisi orientata agli oggetti è basata sulla descrizione del dominio pensata ad oggetti. Gli strumenti utilizzati sono: Modello di Dominio, SSD (Sequence System Diagram) e Contratti delle operazioni.

2.2.1. Modello di Dominio

La Modellazione del Business comprende la stesura del Modello di Dominio ed un elaborato grafico che identifica i concetti, gli attributi e le associazioni significative. Dopo una valutazione dello scenario principale di successo dei casi d'uso UC2, UC3, UC5 e UC8 sono state identificate le seguenti classi concettuali:

- Cliente: È sia attore che entità nel modello. Rappresenta l'utente finale del ristorante, interagisce con il sistema (come attore) per effettuare prenotazioni e altre operazioni.
- MyResto: È il sistema informatico che gestisce tutte le operazioni del ristorante.
- Amministratore: Colui che interagisce con il sistema.

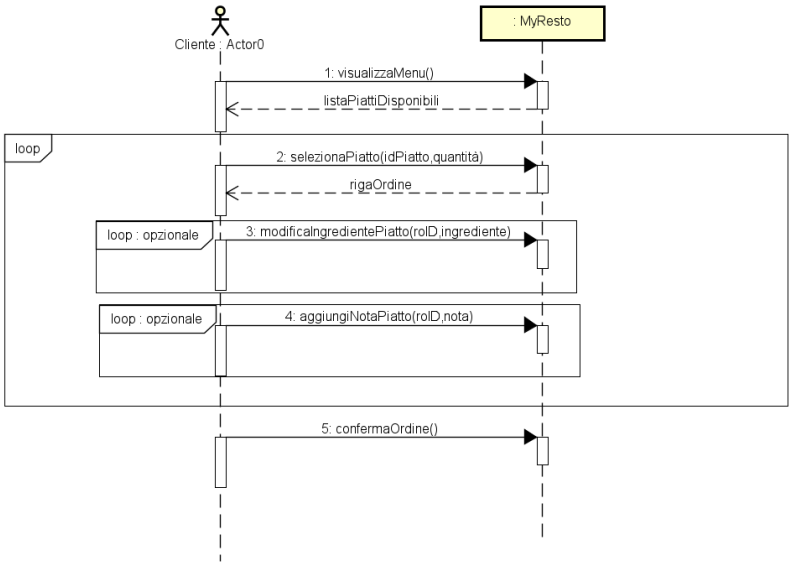
- Ordine: Rappresenta l’insieme dei piatti e bevande richiesti da un cliente a un tavolo in un determinato momento.
- RigaOrdine: Rappresenta una singola voce all'interno di un Ordine, specificando la quantità di un piatto e il suo prezzo al momento dell'ordine.
- Piatto: Voce del menù. Include nome, descrizione, prezzo, ingredienti, allergeni, categoria.
- Menù: contiene l’elenco dei piatti e bevande disponibili.
- Pagamento: Entità che rappresenta il pagamento di un ordine, incluso importo, metodo, data e stato.
- Ingrediente: Rappresenta un singolo componente alimentare utilizzato per comporre un Piatto.
- Variazione: Rappresenta una modifica o un'aggiunta specifica a un ingrediente o a un piatto (es."doppio formaggio").
- Turno: Indica un intervallo temporale di lavoro assegnato ai dipendenti; include data e fascia oraria.
- Dipendente: Rappresenta un membro dello staff del ristorante (es. cameriere, cuoco). Include nome, ruolo, disponibilità e i turni assegnati.
- RiscattaPremio:Registra l'evento in cui un cliente riscatta un premio utilizzando i suoi punti fedeltà.
- FidelityCard: Rappresenta la tessera fedeltà associata a un cliente, che accumula punti in base agli acquisti e permette il riscatto di premi.
- Premio: Rappresenta un regalo che i clienti possono riscattare utilizzando i punti della loro Fidelity Card.
- Prenotazione: Rappresenta la richiesta di un cliente per riservare un tavolo in una specifica data e orario, indicando anche il numero di persone.
- Tavolo: Rappresenta un tavolo fisico nel ristorante. Include informazioni come numero identificativo, numero massimo di posti, posizione (intero o esterno) e stato (libero, occupato, prenotato).



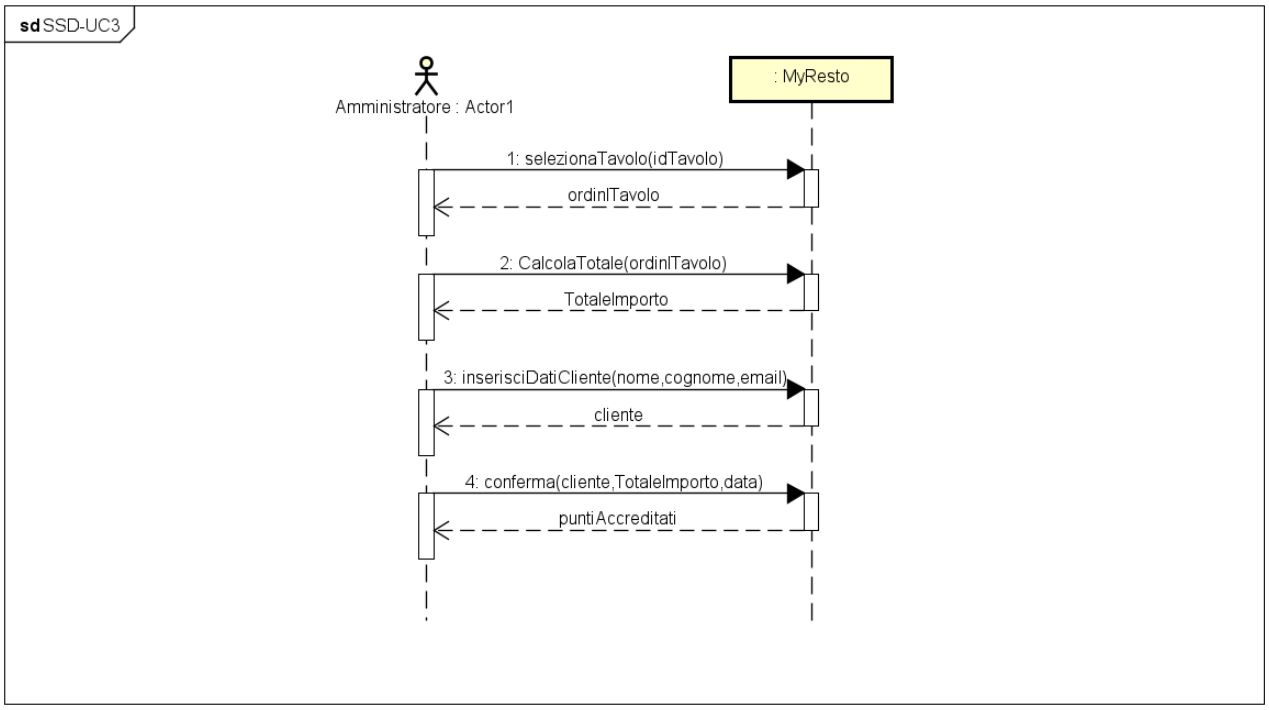
2.2.2. Diagramma di sequenza di sistema

Procedendo con l'analisi Orientata agli Oggetti, il passo successivo è la creazione del Diagramma di Sequenza di Sistema (SSD) al fine di illustrare il corso degli eventi di input e di output per lo scenario principale di successo nei casi d'uso scelti (UC2, UC3, UC5 e UC8), quindi avremo:

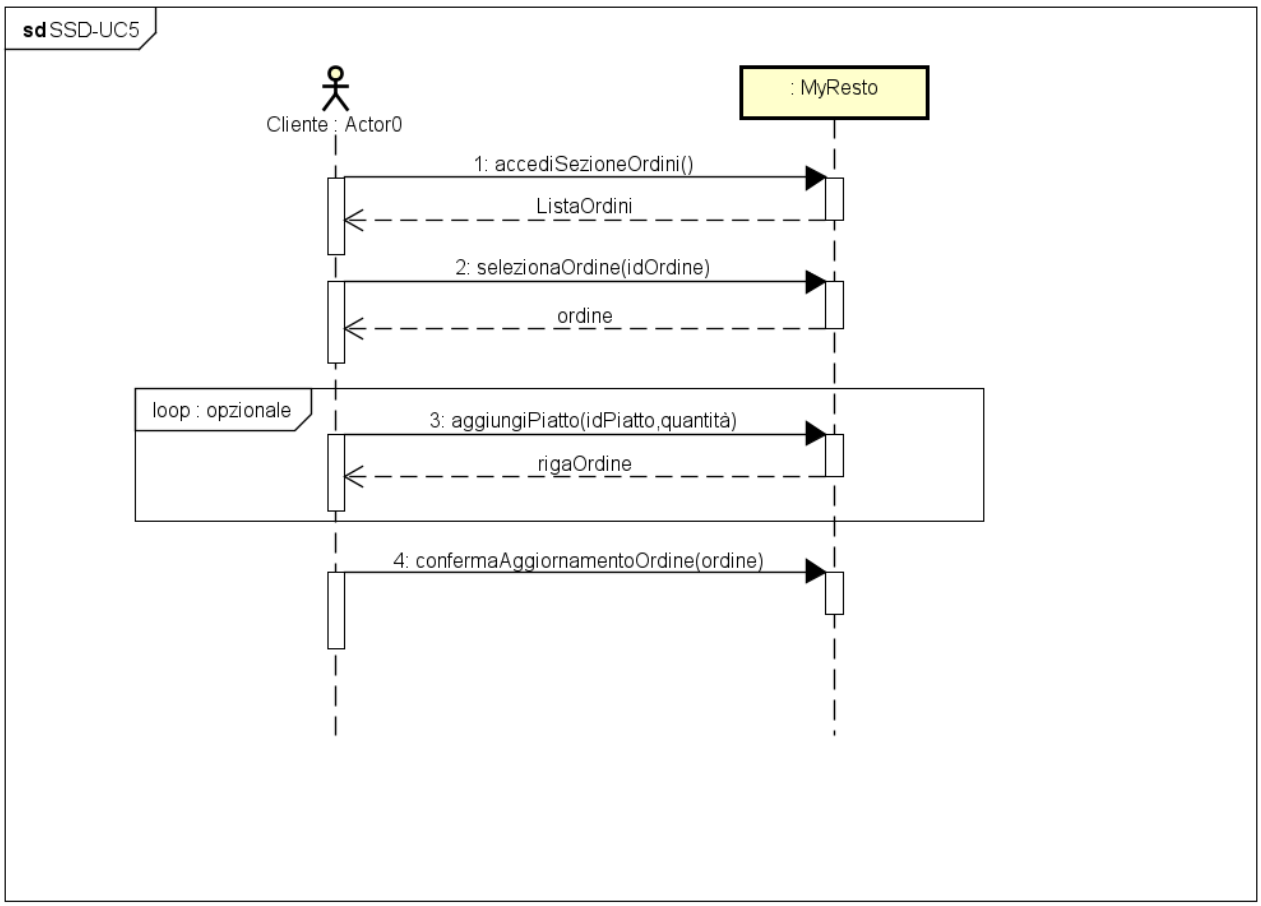
UC2:


Cliente : Actor0
: MyResto
1: visualizzaMenu()
listaPiattiDisponibili
loop
2: selezionaPiatto(idPiatto,quantità)
rigaOrdine
loop : opzionale
3: modificaIngredientePiatto(rolD,ingrediente)
loop : opzionale
4: aggiungiNotaPiatto(rolD,nota)
5: confermaOrdine()

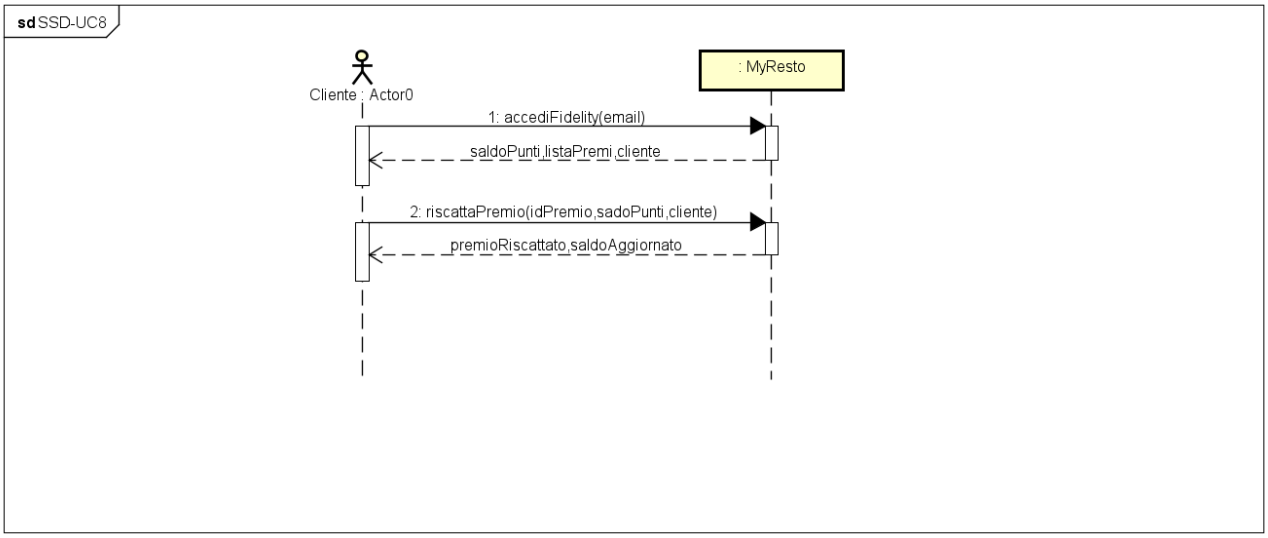
UC3:


sd SSD-UC3
Amministratore : Actor1
: MyResto
1: selezionaTavolo(idTavolo)
ordinlTavolo
2: CalcolaTotale(ordinlTavolo)
TotaleImporto
3: inserisciDatiCliente(nome,cognome,email)
cliente
4: conferma(cliente,TotaleImporto,data)
puntiAccreditati

UC5:


sd SSD-UC5
Cliente : Actor0
: MyResto
1: accediSezioneOrdini()
ListaOrdini
2: selezionaOrdine(idOrdine)
ordine
loop : opzionale
3: aggiungiPiatto(idPiatto,quantità)
rigaOrdine
4: confermaAggiornamentoOrdine(ordine)

UC8:


sd SSD-UC8
Cliente : Actor0
: MyResto
1: accediFidelity(email)
saldoPunti,listaPremi,cliente
2: riscattaPremio(idPremio,sadoPunti,cliente)
premioRiscattato,saldoAggiornato

2.2.3. Contratti delle operazioni

Vengono ora descritte attraverso i Contratti le principali operazioni di sistema che si occupano di gestire gli eventi di sistema individuati nell'SSD.

UC2:

Contratto CO1: visualizzaMenu

Operazione: *visualizzaMenu().*

Riferimenti: caso d'uso: Gestione Ordine.

Pre-condizioni:

- Il cliente ha un tavolo assegnato.
- Il sistema è operativo e il menu è configurato e disponibile.

Post-condizioni:

- Il sistema accede all'istanza m del Menu.
- Vengono recuperati tutti i piatti dalla collezione Piatti:list<Piatto> associata al Menu.
- Una lista di piatti (listaPiatti) è disponibile per essere mostrata al cliente, permettendogli di selezionare i piatti desiderati.

Contratto CO2: selezionaPiatto

Operazione: *selezionaPiatto(idPiatto:int, quantità:int).*

Riferimenti: caso d'uso: Gestione Ordine.

Pre-condizioni:

- Il cliente sta visualizzando il menu.
- L'idPiatto fornito corrisponde a un piatto esistente e disponibile nel Menu.

Post-condizioni:

- Se non c'è nessun ordine in corso viene creata una nuova istanza o di Ordine.
- Se l'ordine è già in corso il sistema recupera l'istanza p di Piatto corrispondente a idPiatto dalla collezione Piatti:list<Piatto>.
- Viene creata una nuova istanza ro di RigaOrdine con il Piatto (p) e la quantità specificata.
- La RigaOrdine creata (ro) viene aggiunta alla collezione righeOrdine:list<RigaOrdine> dell'ordine corrente.

Contratto CO3: modificaIngredientePiatto

Operazione:*modificaIngredientePiatto(roID:int,ingrediente:String,tipoModifica: String)*

Riferimenti: caso d'uso: Gestione Ordine.

Pre-condizioni:

- Il cliente ha un ordine in corso e ha precedentemente selezionato un piatto, creando una RigaOrdine.

Post-condizioni:

- Il sistema recupera l'istanza ro di RigaOrdine corrispondente a roID dalla collezione righeOrdine:list<RigaOrdine>.
- Il sistema recupera l'istanza p di Piatto associato alla RigaOrdine (ro).
- Il sistema recupera l'istanza i di Ingrediente dalla collezione Ingredienti:list<Ingrediente>.
- Viene aggiunta una nuova Variazione (v) alla RigaOrdine (ro) con il tipoModifica e l'ingrediente specificati.
- L'istanza v di Variazione viene creata e aggiunta alla collezione Variazioni:list<Variazione>.

Contratto CO4: aggiungiNota

Operazione: *aggiungiNota(roID:int, nota:String).*

Riferimenti: caso d'uso: Gestione Ordine.

Pre-condizioni:

- Il cliente ha un ordine in corso e ha precedentemente selezionato un piatto, creando una RigaOrdine.

Post-condizioni:

- Il sistema recupera l'istanza ro di RigaOrdine corrispondente a roID dalla collezione righeOrdine:list<RigaOrdine>.
- La nota fornita viene aggiunta e associata alla RigaOrdine (ro).

Contratto CO5: confermaOrdine

Operazione: *confermaOrdine().*

Riferimenti: caso d'uso: Gestione Ordine.

Pre-condizioni:

- Il cliente ha un Ordine in corso e che contiene almeno una RigaOrdine.

Post-condizioni:

- L'istanza dell'ordine corrente viene aggiunta alla collezione ordiniSalvati:list<Ordine>.

UC3:

Contratto CO1: selezionaTavolo

Operazione: *selezionaTavolo(idTavolo:int)*

Riferimenti: caso d'uso: Gestione Pagamento.

Pre-condizioni: L'Amministratore è autenticato nel sistema.

Post-condizioni:

- Il sistema cerca l'ordine associato al idTavolo specificato all'interno della collezione ordiniSalvati:list<Ordine>.
- L'ordine del tavolo (ordiniTavolo) viene recuperato.

Contratto CO2: calcolaTotale

Operazione: *prezzoTot(ordiniTavolo: list<Ordine>)*

Riferimenti: caso d'uso: Gestione Pagamento.

Pre-condizioni: Il sistema ha precedentemente recuperato la lista degli ordini (ordiniTavolo) associati al tavolo selezionato.

Post-condizioni:

- Il sistema itera attraverso ogni Ordine in ordiniTavolo.
- Per ogni Ordine, il sistema recupera la sua listaRO.
- Per ogni ro di RigaOrdine (ro):
    - Viene recuperato il prezzo base del piatto associato a ro.
    - Viene recuperata la listaV (lista di Variazione) associata a ro.
    - Per ogni Variazione in listaV, viene recuperato il costoV della variazione.
    - Il prezzo della RigaOrdine viene aggiornato per includere i costoV delle variazioni.
- - Il sistema calcola il subtotale per ogni Ordine sommando i prezzi di tutte le sue RigaOrdine modificate.
- - Il sistema calcola il totaleImporto complessivo sommando i subtotale di tutti gli Ordine in ordiniTavolo.

Contratto CO3: inserisciCliente

Operazione:*inserisciDatiCliente(nome:String,cognome:String, email:String).*

Riferimenti: caso d'uso: Gestione Pagamento.

Pre-condizioni:

- L'Amministratore ha calcolato il totale dell'ordine e sta procedendo con la gestione del pagamento.

- I dati nome, cognome, email del cliente sono stati forniti dall'amministratore.

Post-condizioni:

- Il sistema cerca un cliente esistente nella collezione Clienti:list<Cliente> utilizzando l'email fornita.
- Se il cliente non esiste:
    - Una nuova istanza cliente di Cliente viene creata con i nome, cognome e email forniti.
    - L'istanza cliente di Cliente viene aggiunta alla collezione Clienti:list<Cliente>.

-Se il Cliente esiste:

- Viene recuperata l'istanza c di Cliente dalla collezione Clienti:list<Cliente> con l'email fornita.

Contratto CO4: conferma

Operazione: *conferma(cliente: Cliente, totaleImporto: float,data: Date)*

Riferimenti: caso d'uso: Gestione Pagamento.

Pre-condizioni:

- Il TotaleImporto è stato precedentemente calcolato.
- I dati del cliente sono stati inseriti o recuperati.

Post-condizioni:

- Viene creata una nuova istanza pa di Pagamento con i parametri cliente, TotaleImporto e data.
- L'istanza pa di Pagamento viene aggiunta alla collezione Pagamenti:list<Pagamento>.
- Se il cliente non è nullo:
    - Il sistema recupera la FidelityCard (fc) associata al cliente.
    - Il sistema ottiene i punti correnti dalla FidelityCard (fc).
    - I punti della FidelityCard (fc) vengono aggiornati aggiungendo TotaleImporto al saldo esistente.
    - L'istanza del cliente viene aggiornata nel sistema per riflettere il nuovo saldo punti.

UC5:

Contratto CO1: accediSezioneOrdini

Operazione: *accediSezioneOrdini()*

Riferimenti: caso d'uso: Modifica ordine già effettuato.

Pre-condizioni: Il Cliente ha effettuato l'accesso al sistema.

Post-condizioni:

- ll sistema recupera tutti gli ordini dalla collezione ordiniSalvati:list<Ordine>.
- Una lista di ordini (listaOrdini) è disponibile per essere visualizzata dal cliente, permettendogli di selezionare l'ordine da modificare.

Contratto CO2: selezioneOrdine

Operazione: *selezionaOrdine(idOrdine: int)*

Riferimenti: caso d'uso: Modifica ordine già effettuato.

Pre-condizioni: Il cliente ha precedentemente visualizzato una lista di ordini disponibili per la modifica.

Post-condizioni:

- Il sistema cerca l'istanza dell'Ordine (ordine) corrispondente a idOrdine all'interno della collezione ordiniSalvati:list<Ordine>.
- L'istanza dell'Ordine selezionato (ordine) viene recuperata e resa disponibile al sistema.

Contratto CO3: confermaAggiornamentoOrdine

Operazione:*confermaAggiornamentoOrdine(ordine: Ordine).*

Riferimenti: caso d'uso: Modifica ordine già effettuato.

Pre-condizioni: Un ordine (ordine) è stato precedentemente selezionato dal cliente per la modifica.

Post-condizioni:L'istanza ordine viene aggiornata nella collezione ordiniSalvati:list<Ordine>.

UC8:

Contratto CO1: accediFidelity

Operazione: *accediFidelity(email:String)*

Riferimenti: caso d'uso: Fidelity Card.

Pre-condizioni: Il cliente è registrato nel sistema.

Post-condizioni:

- Il sistema cerca l'istanza c del Cliente corrispondente all'email fornita all'interno della collezione Clienti:list<Cliente>.
- Se il cliente (c) viene trovato:
    - Il sistema recupera l'istanza fc della FidelityCard associata al cliente.
    - Viene recuperato il saldo punti corrente dalla FidelityCard.
    - Viene recuperata la listaPremi disponibile dalla collezione Premi:list<Premio>.

Contratto CO2: riscattaPremio

Operazione: *riscattaPremio(idPremio:int, saldoPunti:int, cliente:Cliente)*

Riferimenti: caso d'uso: Fidelity Card.

Pre-condizioni: Il cliente ha effettuato l'accesso alla sezione Fidelity Card.

Post-condizioni:

- Il sistema recupera l'istanza prem di Premio corrispondente a idPremio dalla collezione Premi:list<Premio>.
- Il sistema recupera il costoPunti del Premio (prem).
- Viene creata una nuova istanza pr di PremioRiscattato con il cliente, il premio e la dataRiscatto corrente.
- L'istanza pr di PremioRiscattato viene aggiunta alla collezione premiRiscattati:list<PremioRiscattato>.
- I punti sulla FidelityCard del cliente vengono aggiornati, sottraendo il costoPunti.
- L'istanza del cliente viene aggiornata nel sistema per riflettere il nuovo saldo punti.

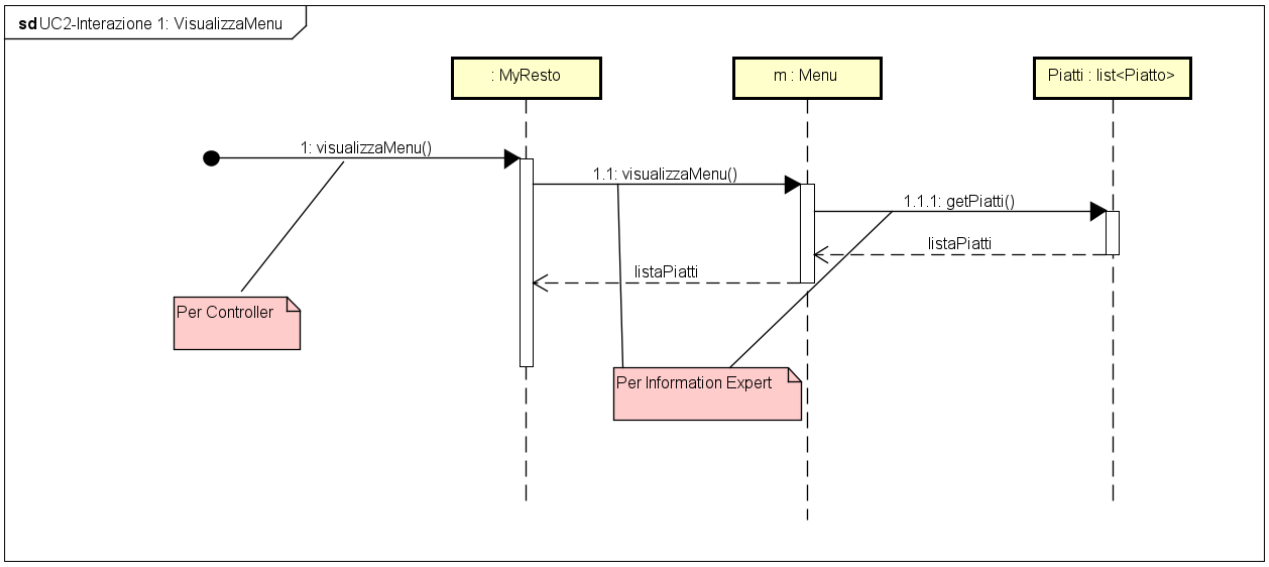
2.3 Progettazione

La progettazione orientata agli oggetti è la disciplina di UP interessata alla definizione degli oggetti software, delle loro responsabilità e a come questi collaborano per soddisfare i requisiti individuati nei passi precedenti. L'elaborato principale di questa fase, che è stato preso in considerazione, è il **Modello di Progetto**, ovvero l'insieme dei diagrammi che descrivono la progettazione logica sia da un punto di vista dinamico (Diagrammi di Interazione) che da un punto di vista statico (Diagrammi delle Classi). Seguono dunque i diagrammi di interazione più significativi e il diagramma delle classi relativi ai casi d'uso UC2, UC3, UC5 e UC8 determinati a seguito di un attento studio degli elaborati scritti in precedenza.

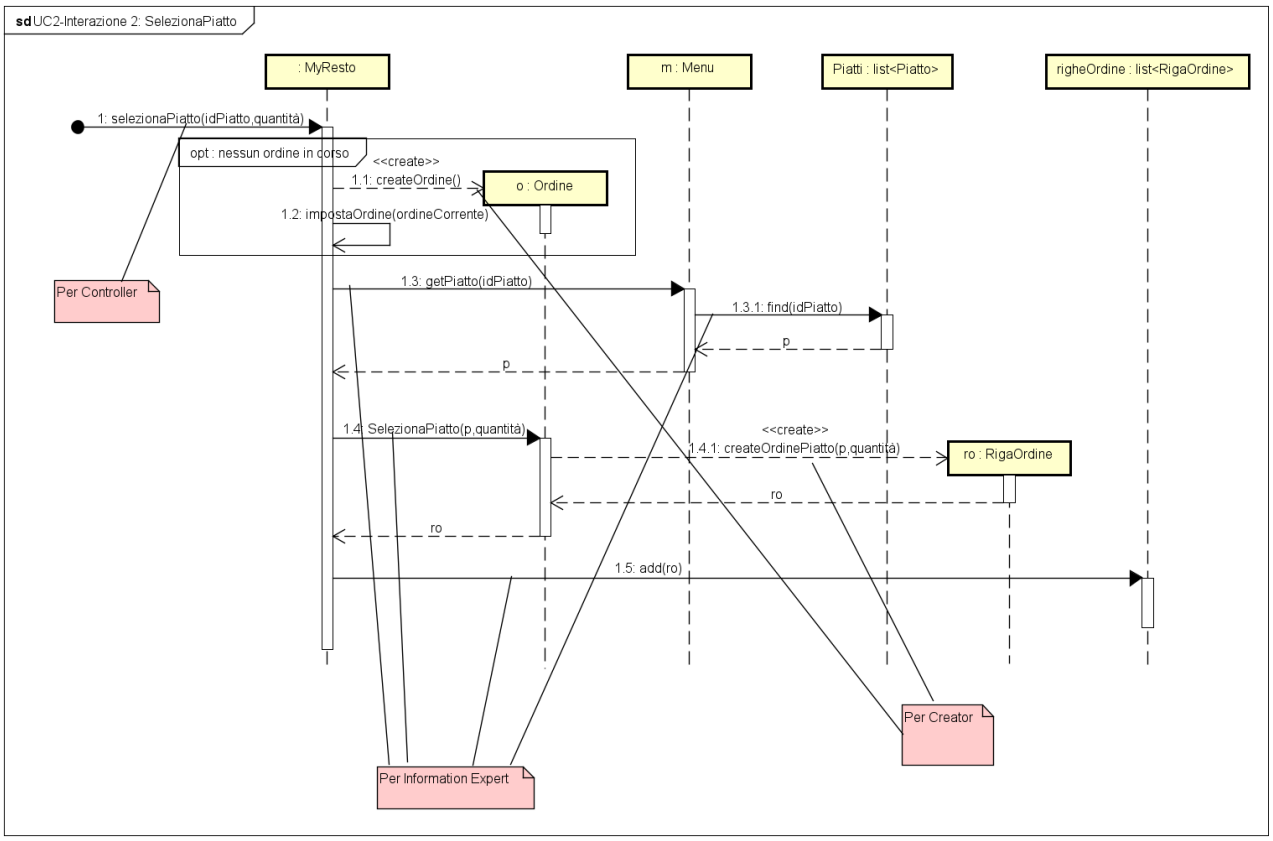
2.3.1 Diagrammi di Sequenza

UC2:

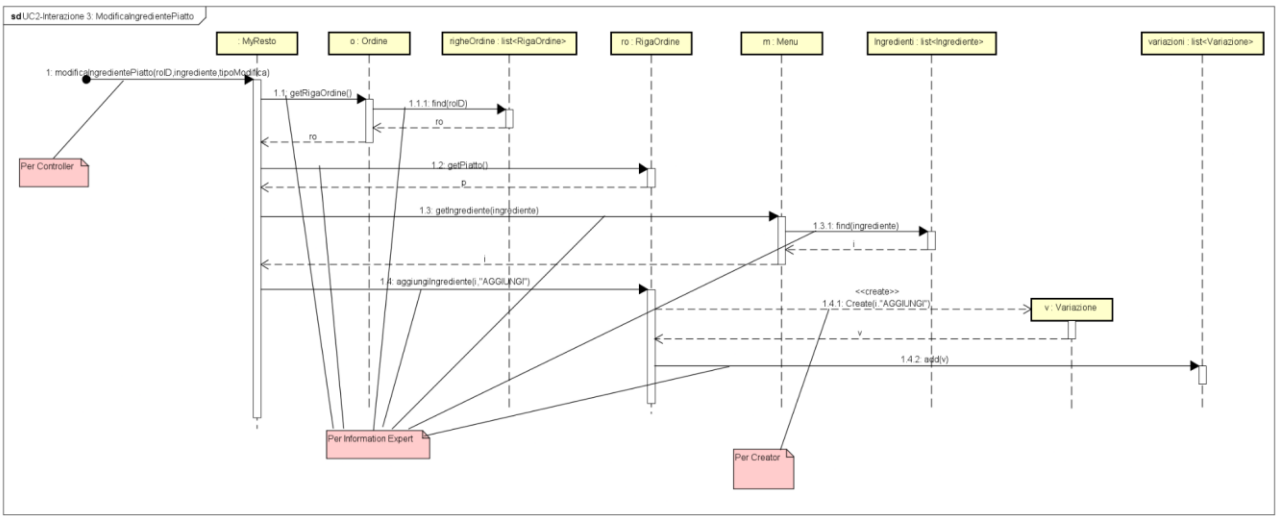
- ***visualizzaMenu***



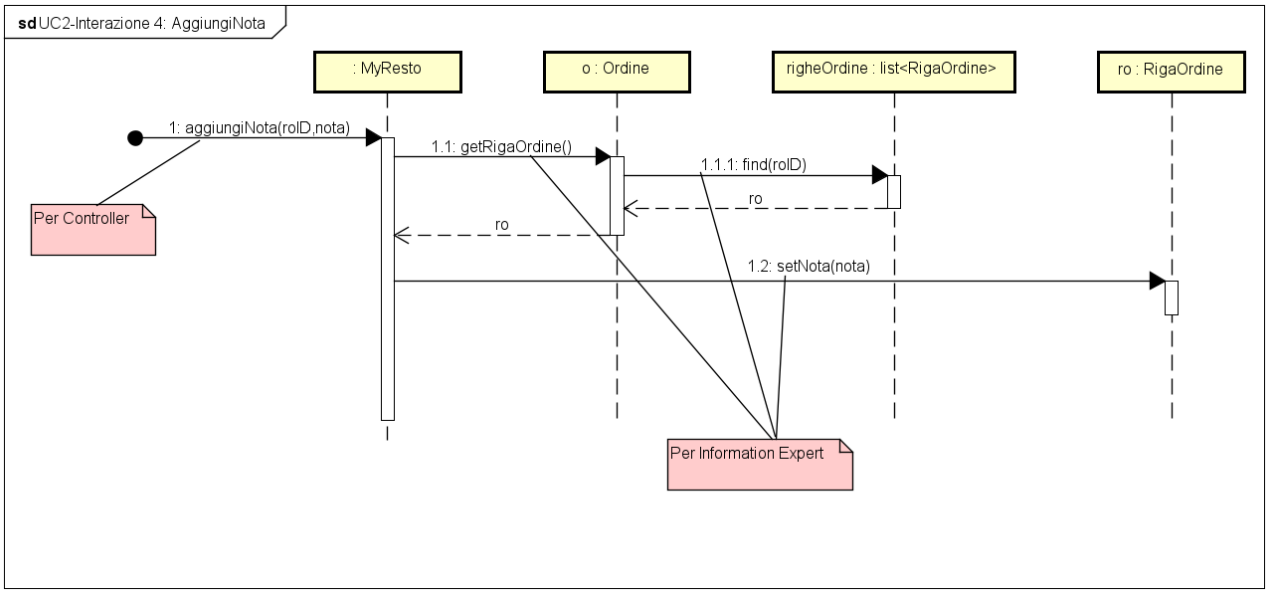
- ***selezionaPiatto***



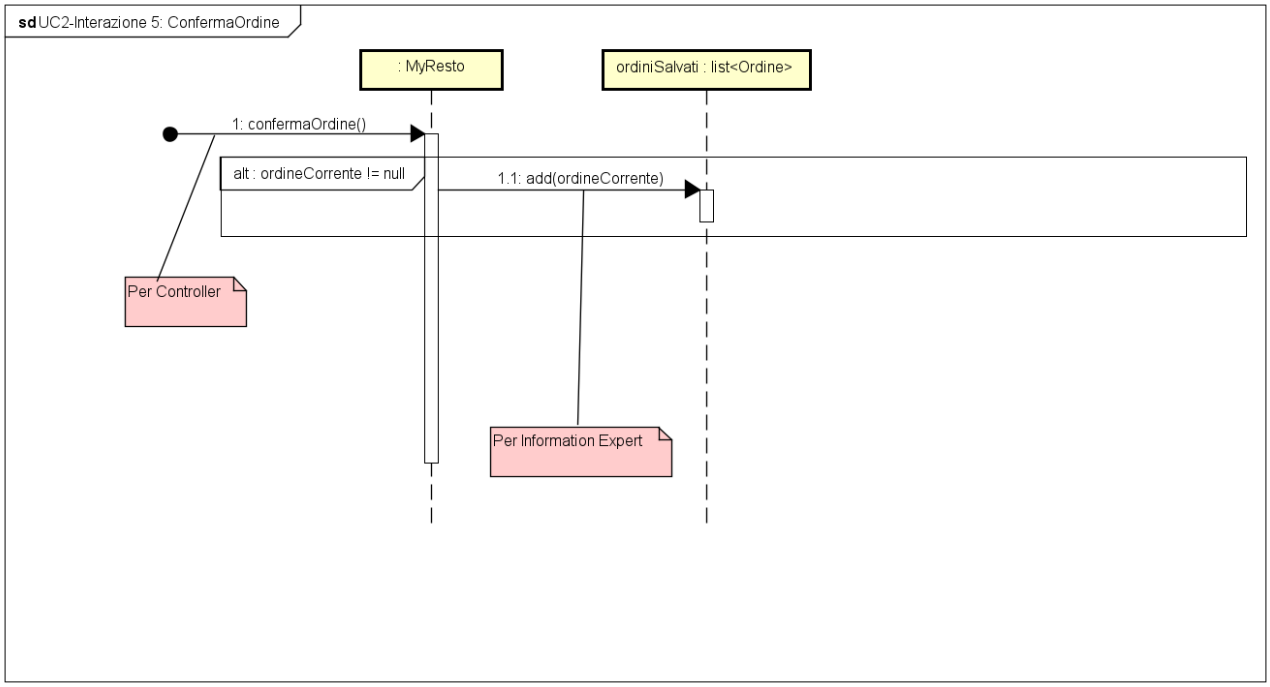
- ***modificaIngredientePiatto***



- ***aggiungiNota***

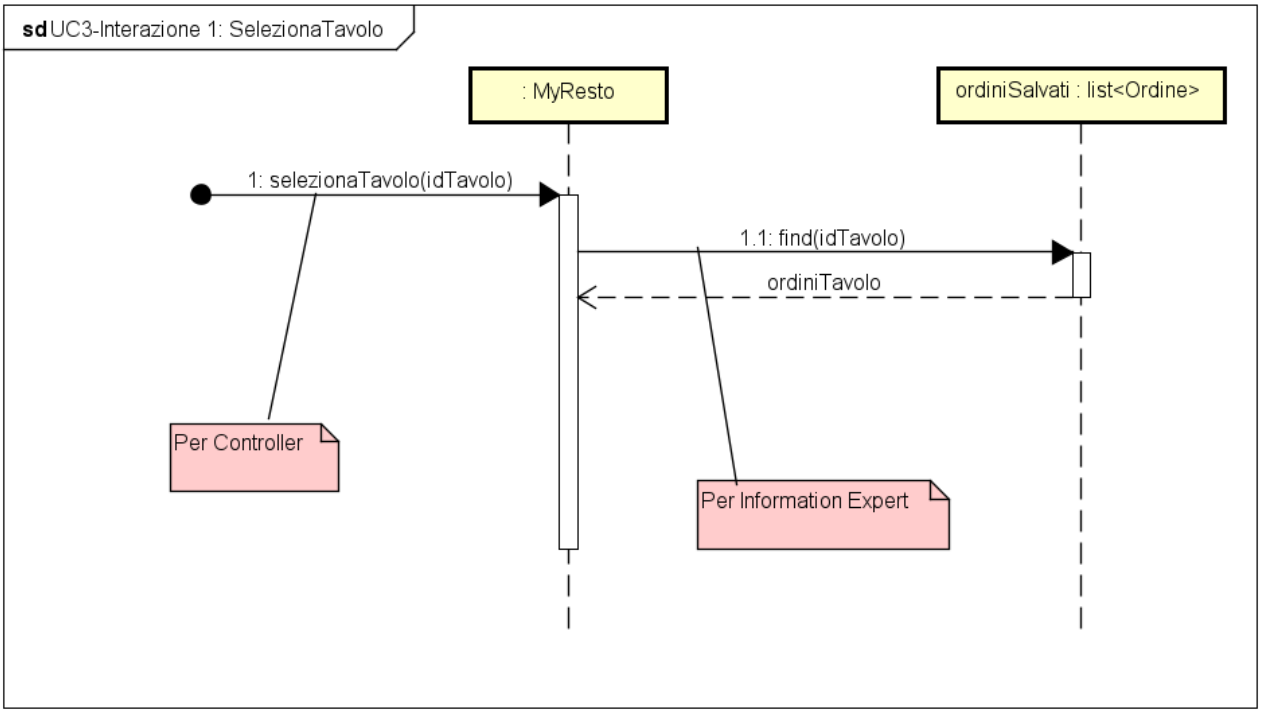


- ***confermaOrdine***

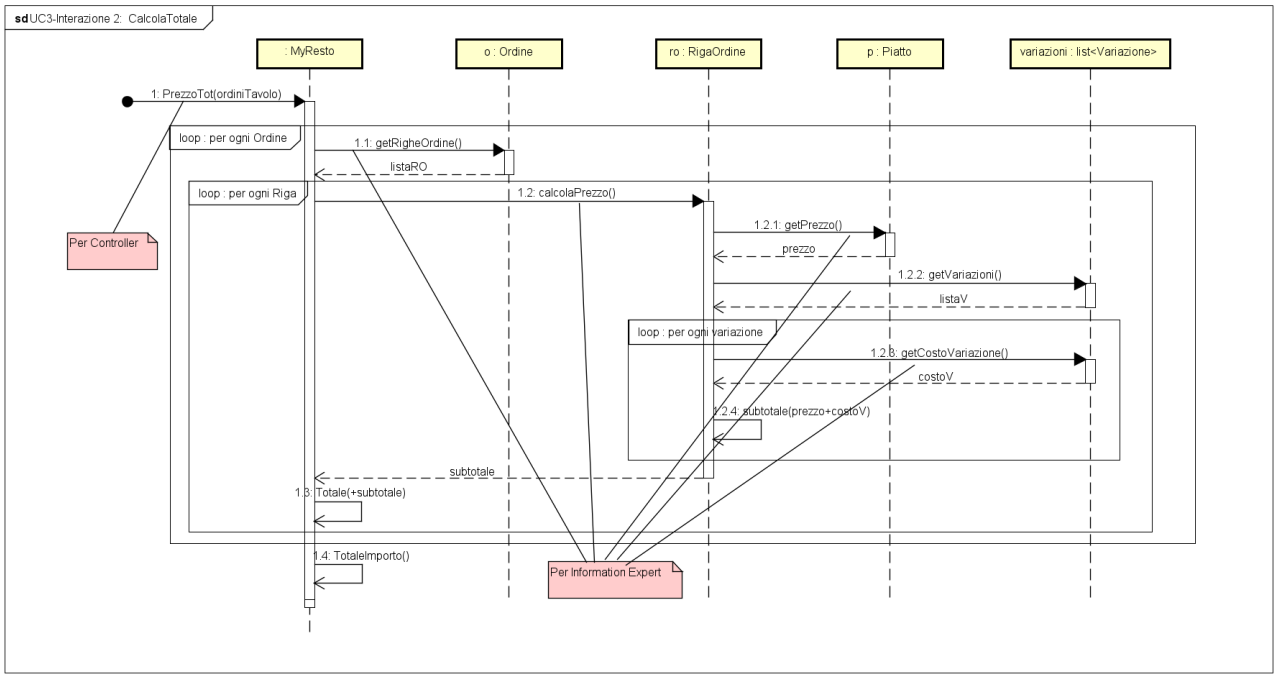


UC3:

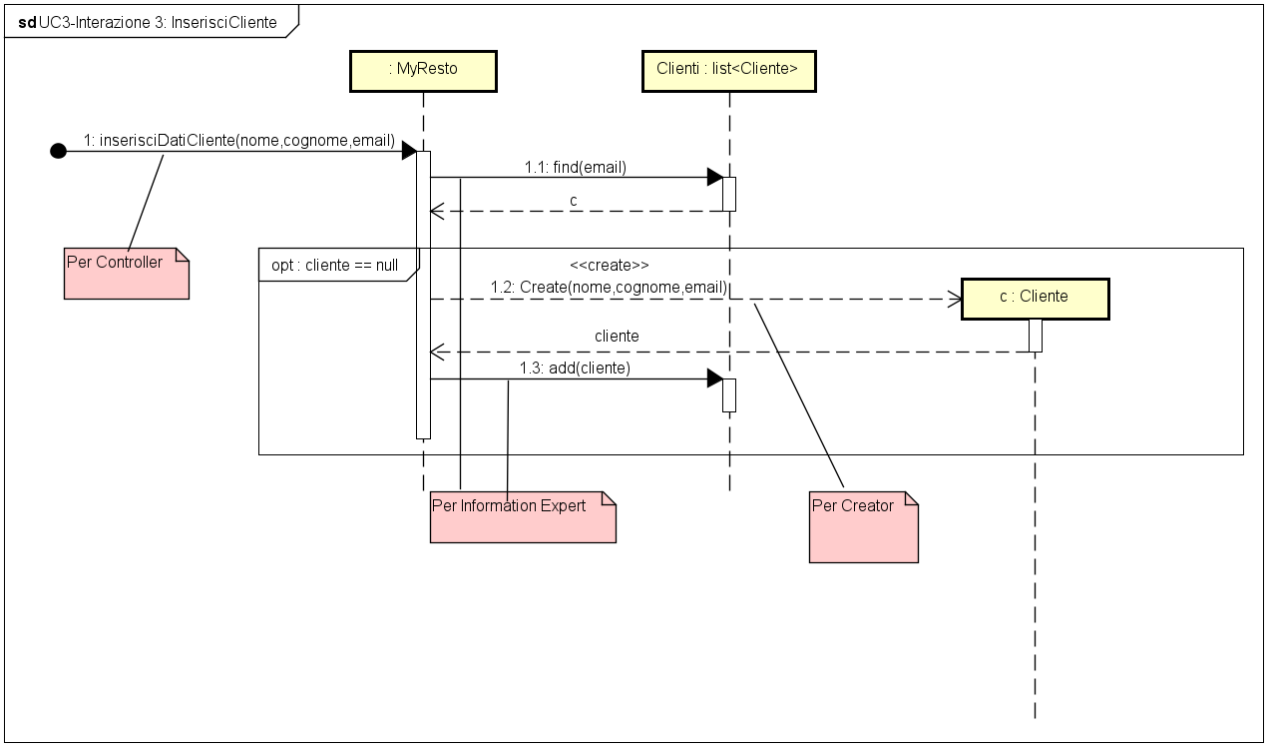
- ***selezionaTavolo***



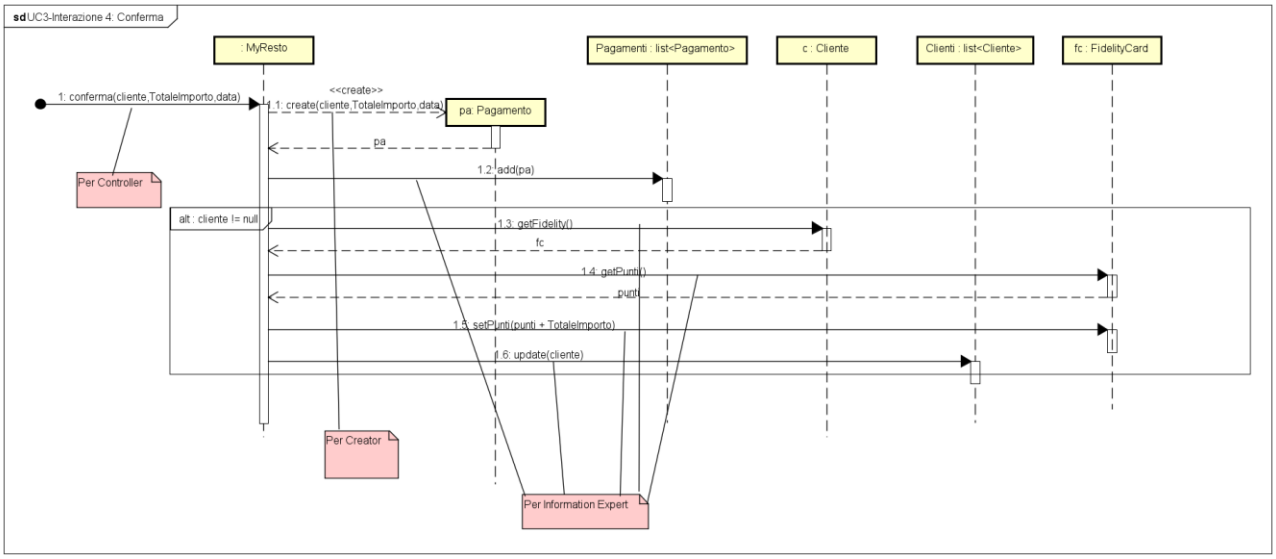
- ***calcolaTotale***



- ***inserisciCliente***

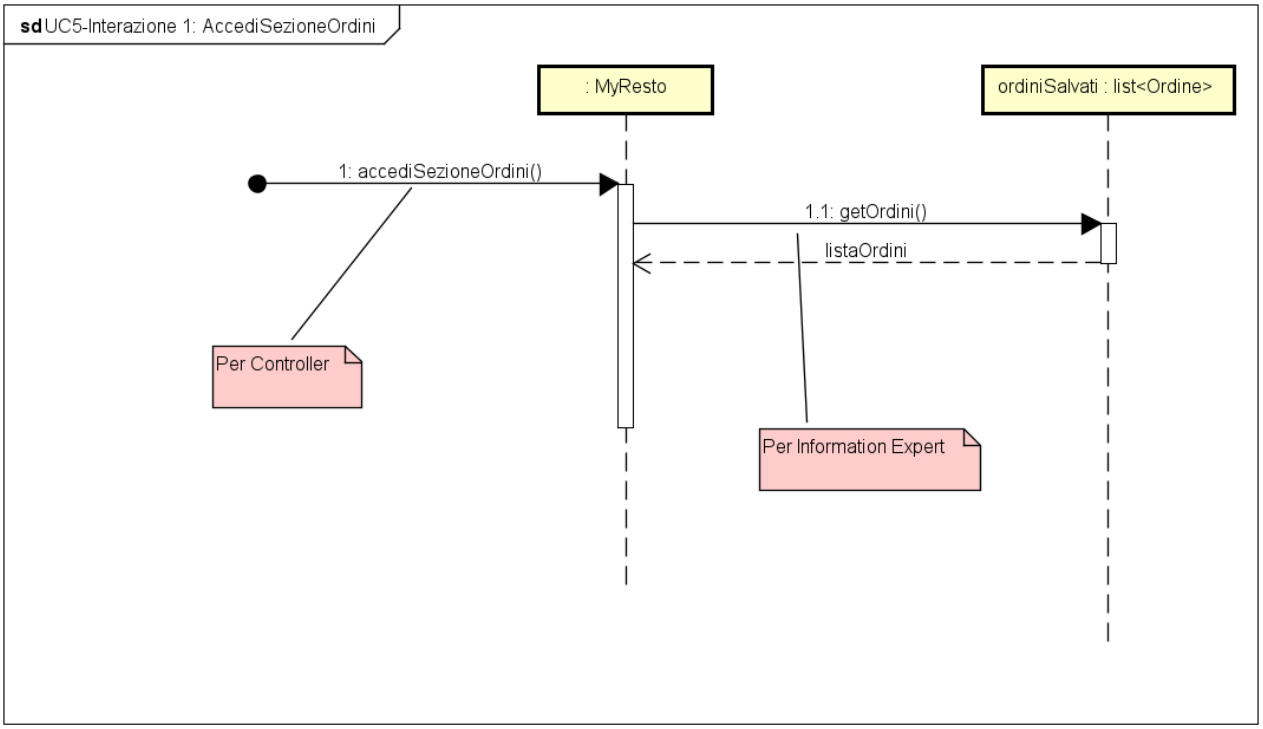


- ***conferma***

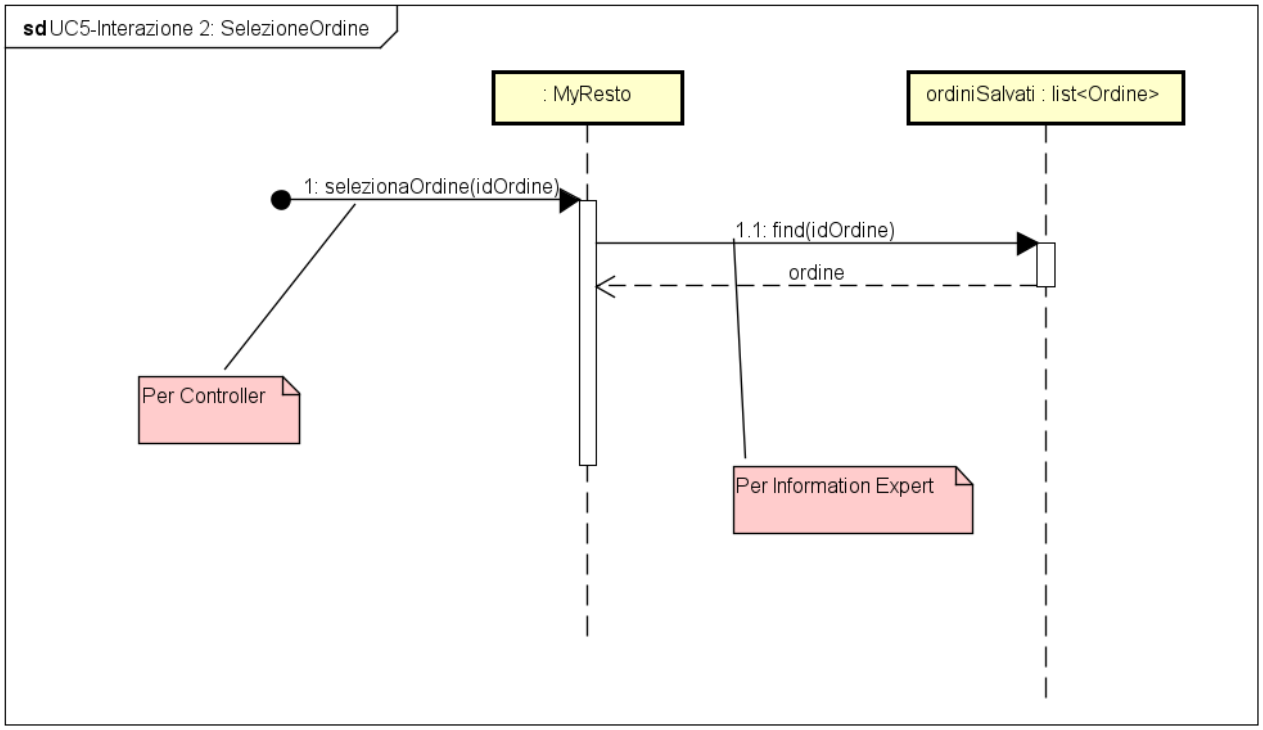


UC5:

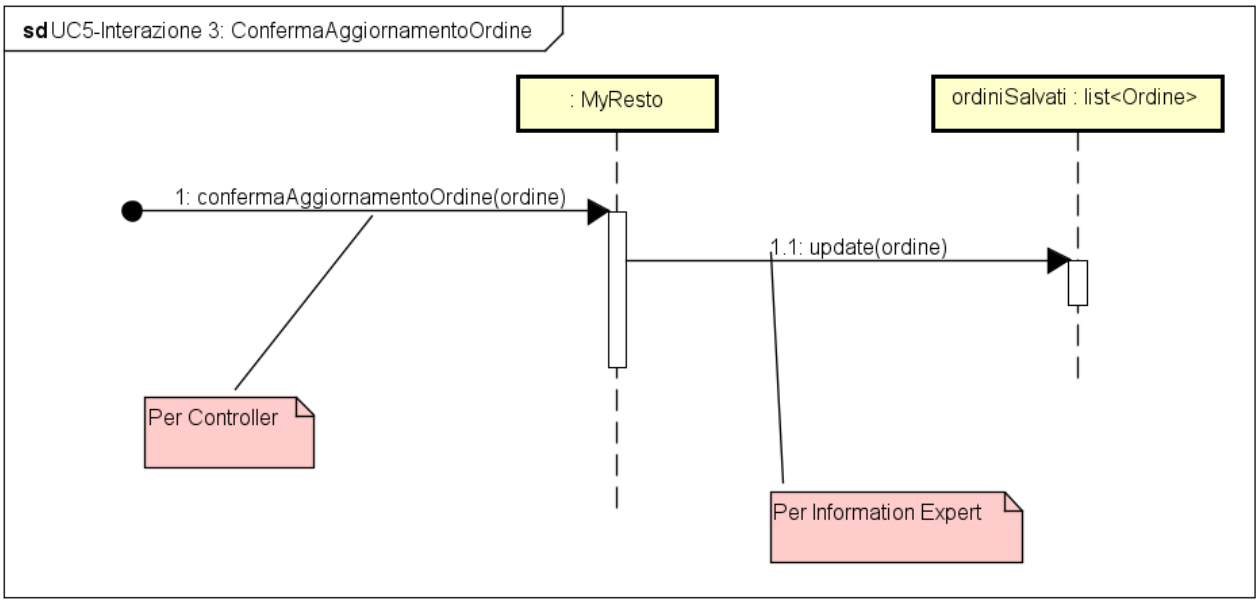
- ***accediSezioneOrdini***



- ***selezioneOrdine***

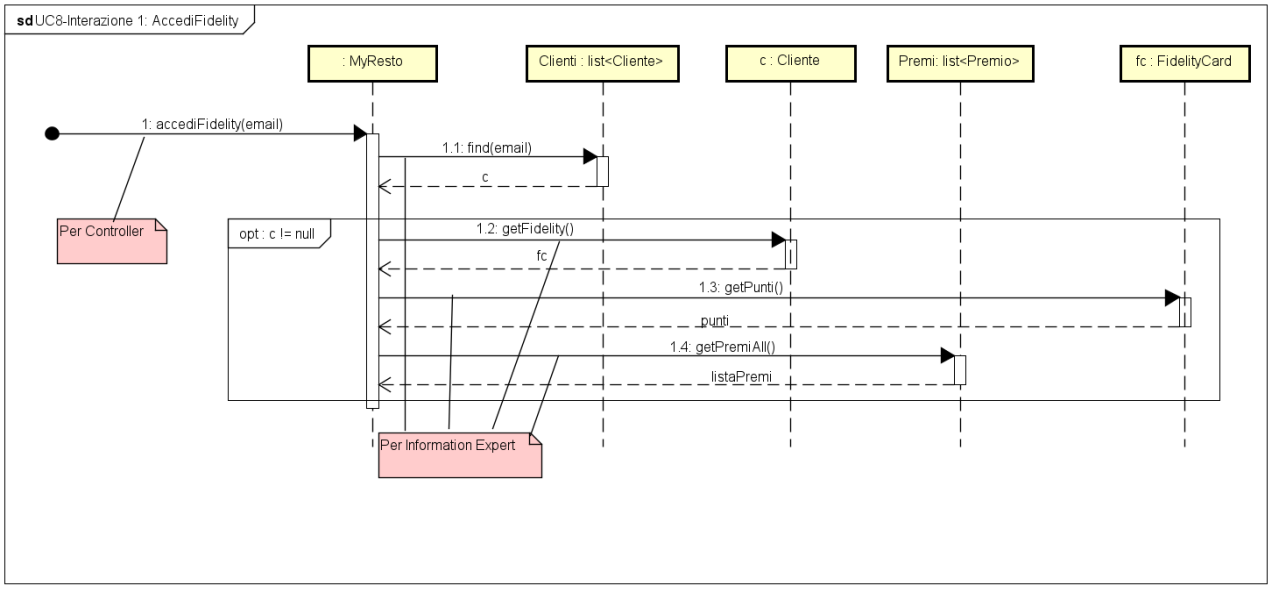


- ***confermaAggiornamentoOrdine***

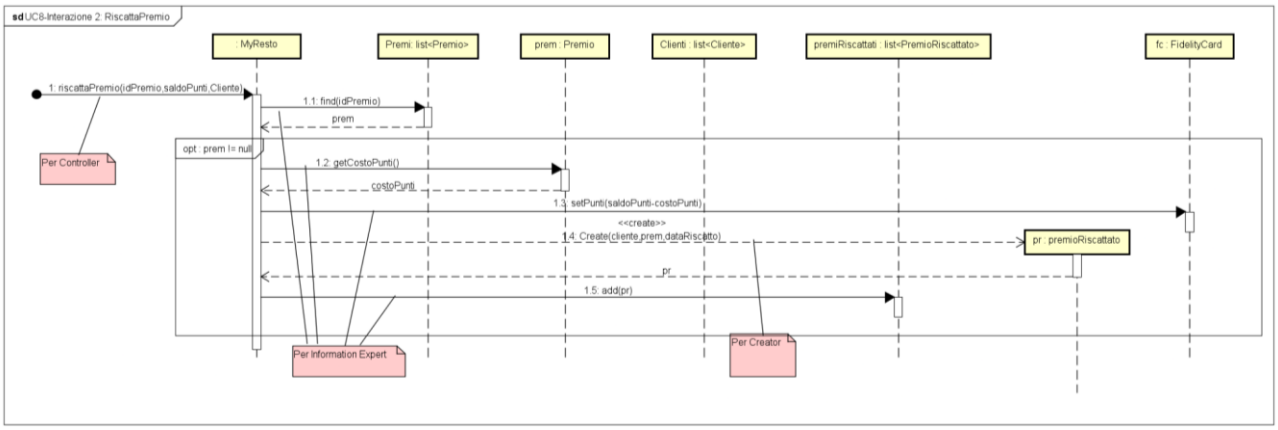


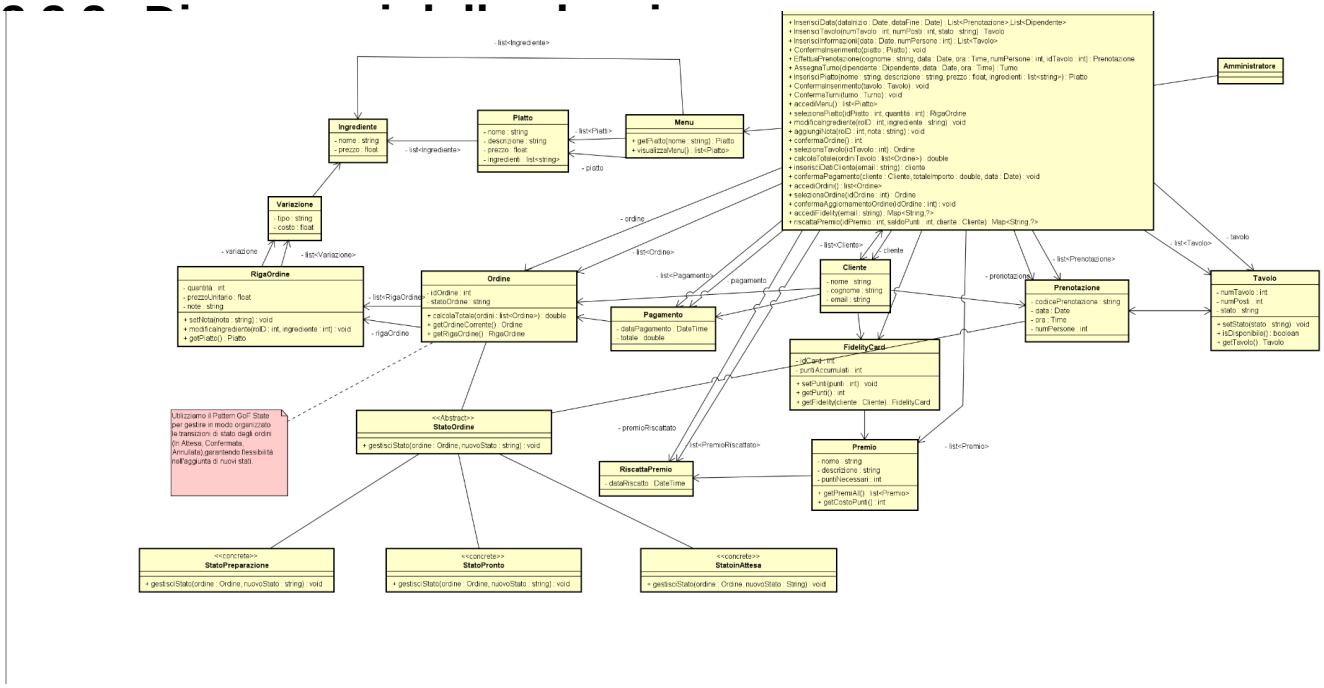
UC8:

- ***accediFidelity***



- ***riscattaPremio***





Abbiamo utilizzato il pattern GoF State per gestire il cambiamento di stato degli ordini nel sistema.

Questa scelta è motivata dai seguenti vantaggi:

- Separazione delle logiche: Il pattern state permette di incapsulare il comportamento associato a ogni stato dell'ordine ("In Attesa","Pronto","In Preparazione") in classi distinte.
- Estensibilità: Aggiungere nuovi stati (es "Annullato") richiede solo l'implementazione di una nuova classe State.