Documento iterazione 3

In base ai documenti stilati prima il team ha deciso di specificare ulteriori casi d' uso in forma dettagliata al fine di includere altre funzionalità nel sistema.

Gli obbiettivi prefissati per questa iterazione sono:

- Approfondimento dei casi d' uso 6,7

1- Modello dei casi d'uso

Sono stati descritti in formato dettagliato i seguenti casi d'uso:

UC6: Ricerca Caterings Cliente

Portata	Applicativo per agenzia di catering
Livello	Obiettivo utente
Attore primario	Amministratore
Parti interessate e interessi	- *Amministratore*: Vuole un sistema pratico e intuitivo per ricercare i catering dei clienti, in modo da controllare le relative informazioni e esercitare in modo più efficace il suo compito. - *Cliente*: Vuole che venga tenuta traccia dei catering effettuati, in particolar modo nel caso in cui questo possa tradursi in eventuali sconti o promozioni.
Pre-condizioni	L'amministratore è identificato e autenticato.
Garanzie di successo	Vengono visualizzati i caterings del cliente
Scenario principale di successo	1. L'Amministratore vuole visionare i caterings del cliente (per esempio durante l'emissione di una fattura per determinare lo sconto). 2. L'Amministratore richiede di visionare i caterings. 3. Il Sistema richiede l'inserimento di nome e cognome del cliente di cui visualizzare i caterings. 4. L'Amministratore inserisce i dati del cliente. 5. Il Sistema mostra i dati del cliente selezionato e chiede conferma. 6. L'Amministratore conferma l'identità del cliente. 7. Il Sistema visualizza i caterings del cliente richiesto.
Scenari alternativi	*a: In qualsiasi momento, il Sistema fallisce: In quanto non vi sono azioni che agiscono su dati persistenti, in questo d'uso un fallimento è più tollerabile in quanto comporterebbe solo un breve ritardo.

	1. L'Amministratore riavvia il Sistema e si autentica. 2. L'Amministratore ricomincia il caso d'uso dal punto 1.
Requisiti speciali	- Interfaccia grafica semplice ed intuitiva (anche se minimale). - Tempi di risposta accettabili.
Elenco delle varianti tecnologiche e dei dati	Per ciascun catering devono essere visualizzati i campi: codice, numero partecipanti, luogo, data, prezzo.
Frequenza di ripetizione	Può essere molto variabile: può avvenire prima di effettuare una fattura, durante una prenotazione, ecc.
Problemi aperti	- Ordinare i dati in ordine temporale.

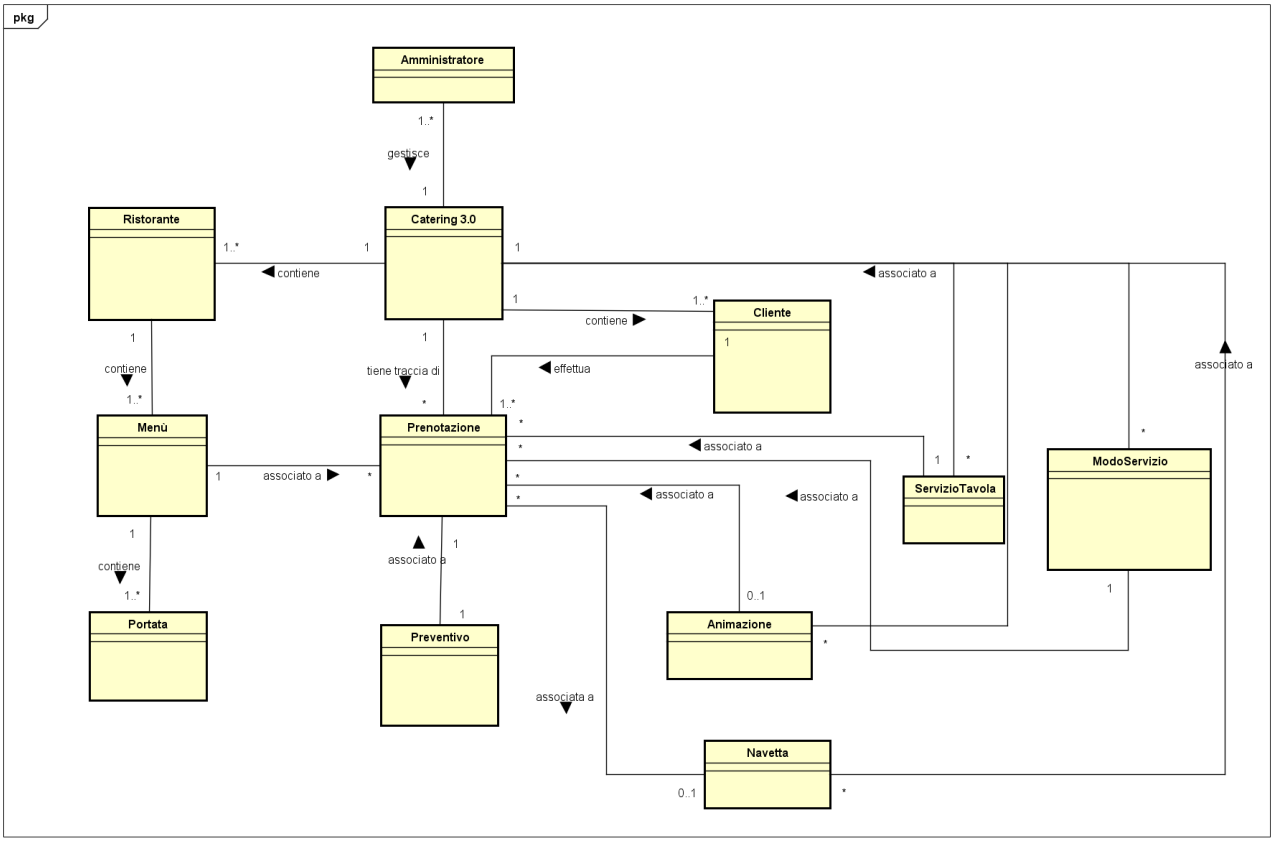
UC7: Ricerca Caterings Ristorante

Portata	Applicativo per agenzia di catering
Livello	Obiettivo utente
Attore primario	Amministratore
Parti interessate e interessi	- *Amministratore*: Vuole un sistema pratico e intuitivo per ricercare i catering organizzati dai ristoranti, in modo da controllare le relative informazioni e esercitare in modo più efficace il suo compito. - *Ristorante*: Vuole che venga tenuta traccia dei catering effettuati, per fini burocratici.
Pre-condizioni	L'amministratore è identificato e autenticato.
Garanzie di successo	Vengono visualizzati i caterings del ristorante.
Scenario principale di successo	1. L'Amministratore vuole visionare i caterings del ristorante (per esempio durante una prenotazione). 2. L'Amministratore richiede di visionare i caterings. 3. Il Sistema richiede l'inserimento del nome e della PIVA del ristorante di cui visualizzare i caterings. 4. L'Amministratore inserisce i dati del ristorante. 5. Il Sistema mostra i dati del ristorante selezionato e chiede conferma. 6. L'Amministratore conferma l'identità del ristorante. 7. Il Sistema visualizza i caterings del ristorante richiesto.
Scenari alternativi	*a: In qualsiasi momento, il Sistema fallisce: In quanto non vi sono azioni che agiscono su dati persistenti, in questo d'uso un fallimento è più tollerabile in quanto comporterebbe solo un breve ritardo. 8. L'Amministratore riavvia il Sistema e si autentica. 9. L'Amministratore ricomincia il caso d'uso dal punto 1.

Requisiti speciali	- Interfaccia grafica semplice ed intuitiva (anche se minimale). - Tempi di risposta accettabili.
Elenco delle varianti tecnologiche e dei dati	Per ciascun catering devono essere visualizzati i campi: codice, numero partecipanti, luogo, data, prezzo.
Frequenza di ripetizione	Può essere molto variabile: può avvenire prima di effettuare una fattura, durante una prenotazione, ecc.
Problemi aperti	- Ordinare i dati in ordine temporale.

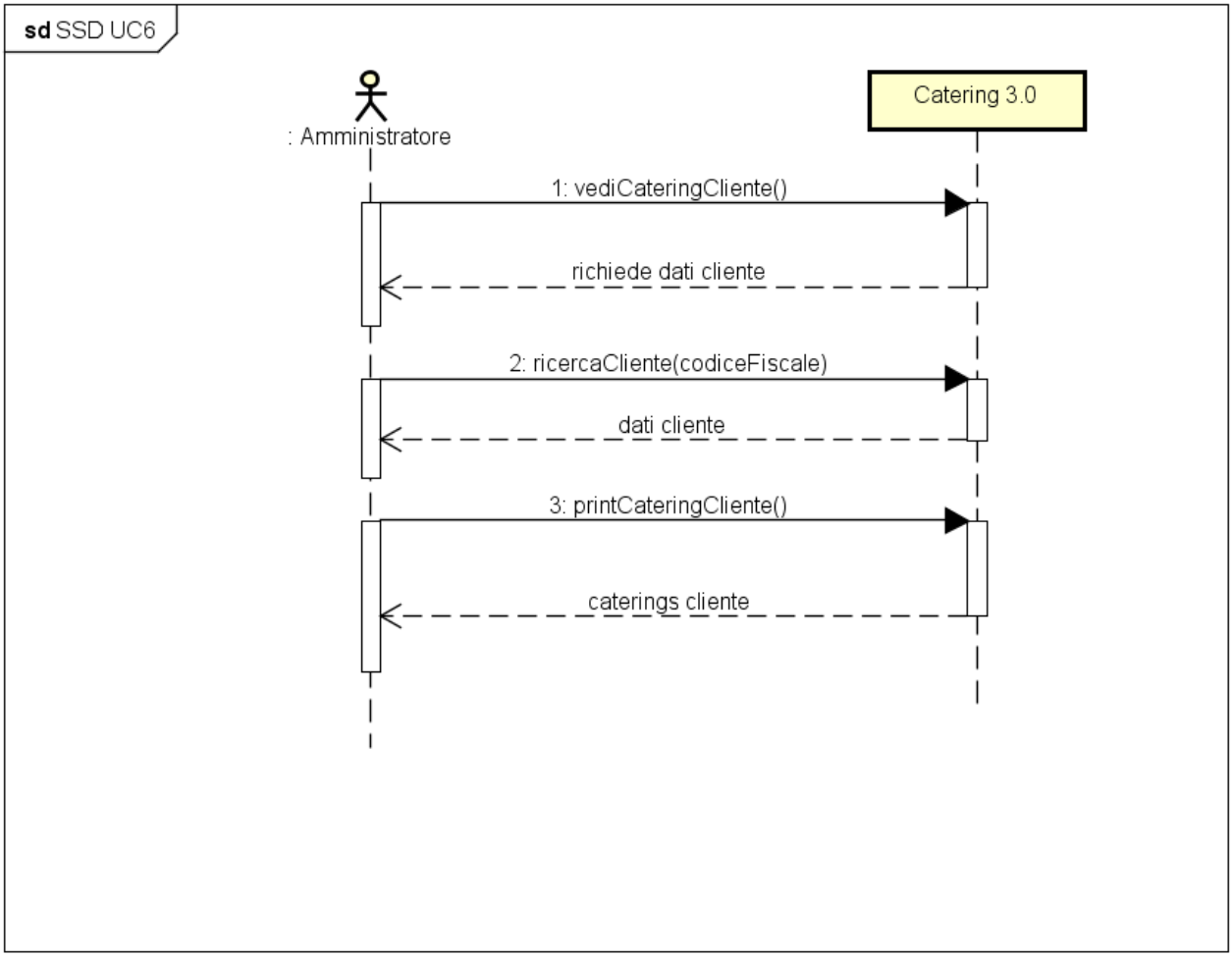
2 – Fase di elaborazione

2.1-Modello di Dominio

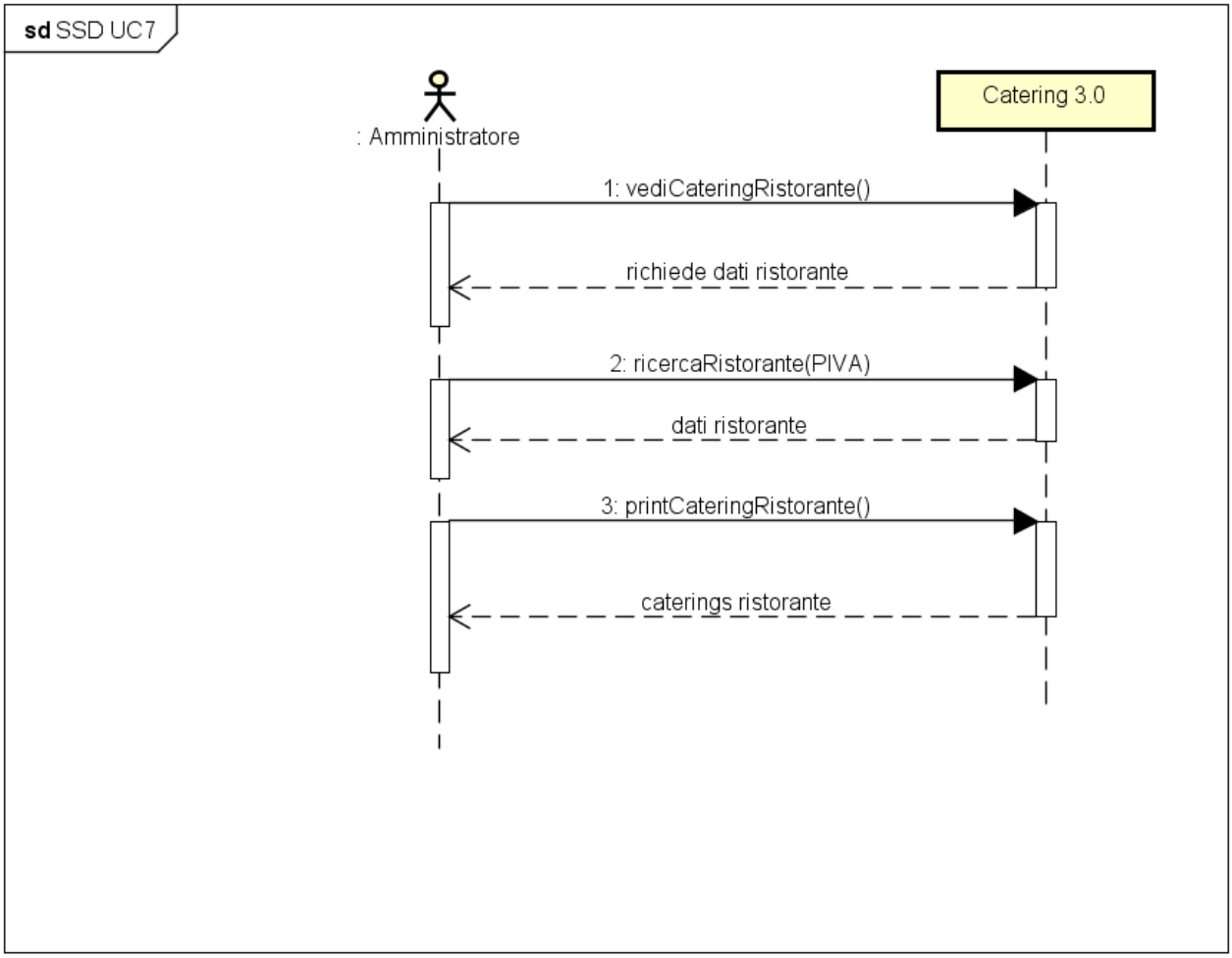


2.2 - Diagramma di sequenza di sistema(SSD)

2.2.1 - SSD Caso d’uso UC6



2.2.2 - SSD Caso d’uso UC7



2.3 - Contratti Operazioni

Dagli SSD è emersa la necessità di definire le seguenti operazioni.

2.3.1-Contratti Operazioni UC6

Contratto CO1: visualizzazioneCatering

Operazione	visualizzazioneCatering()
Riferimenti	Caso d'Uso UC6: Ricerca Caterings Cliente
Pre-Condizioni	Nessuna
Post-Condizioni	- il Sistema richiede l'inserimento dei dati del cliente di cui si vogliono visualizzare i caterings.

Contratto CO2: ricercaCliente

Operazione	ricercaCliente(cf: string)
Riferimenti	Caso d'Uso UC6: Ricerca Caterings Cliente
Pre-Condizioni	- è in corso la ricerca dei caterings di un cliente
Post-Condizioni	- il Sistema visualizza i dati del cliente di cui si vogliono visualizzare i caterings e ne chiede conferma.

Contratto CO3: findCateringbyClient

Operazione	findCateringbyClient()
Riferimenti	Caso d'Uso UC6: Ricerca Caterings Cliente
Pre-Condizioni	- è in corso la ricerca dei caterings di un cliente
Post-Condizioni	- il Sistema visualizza i caterings del cliente selezionato precedentemente.

2.3.2-Contratti Operazioni UC7

Contratto CO1: visualizzazioneCatering

Operazione	visualizzazioneCatering()
Riferimenti	Caso d'Uso UC7: Ricerca Caterings Ristorante
Pre-Condizioni	Nessuna
Post-Condizioni	- il Sistema richiede l'inserimento dei dati del ristorante di cui si vogliono visualizzare i caterings.

Contratto CO2: ricercaRistorante

Operazione	ricercaRistorante(PIVA: string)
Riferimenti	Caso d'Uso UC7: Ricerca Caterings Ristorante
Pre-Condizioni	- è in corso la ricerca dei caterings di un ristorante
Post-Condizioni	- il Sistema visualizza i dati del ristorante di cui si vogliono visualizzare i caterings e ne chiede conferma.

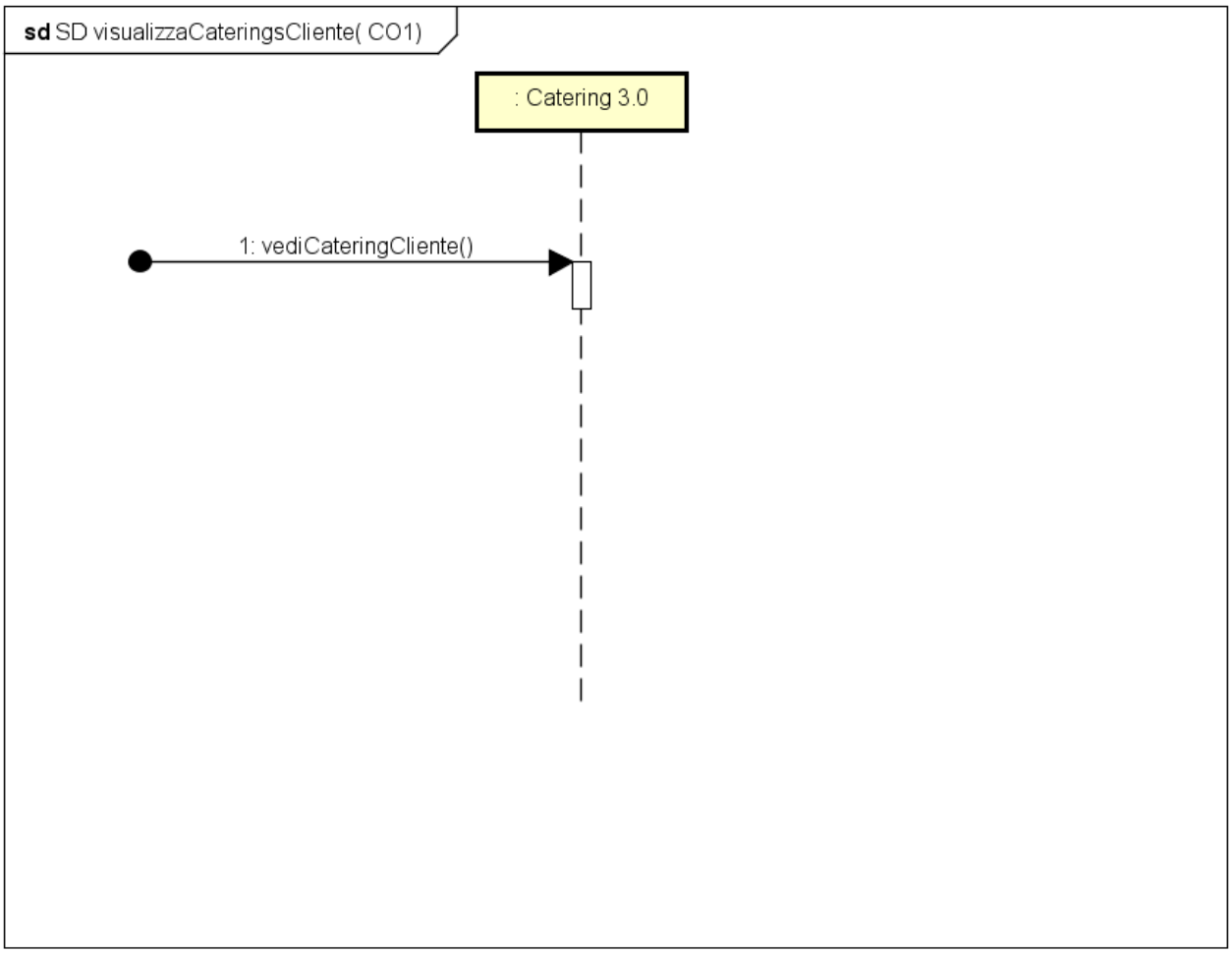
Contratto CO3: findCateringbyRestaurant

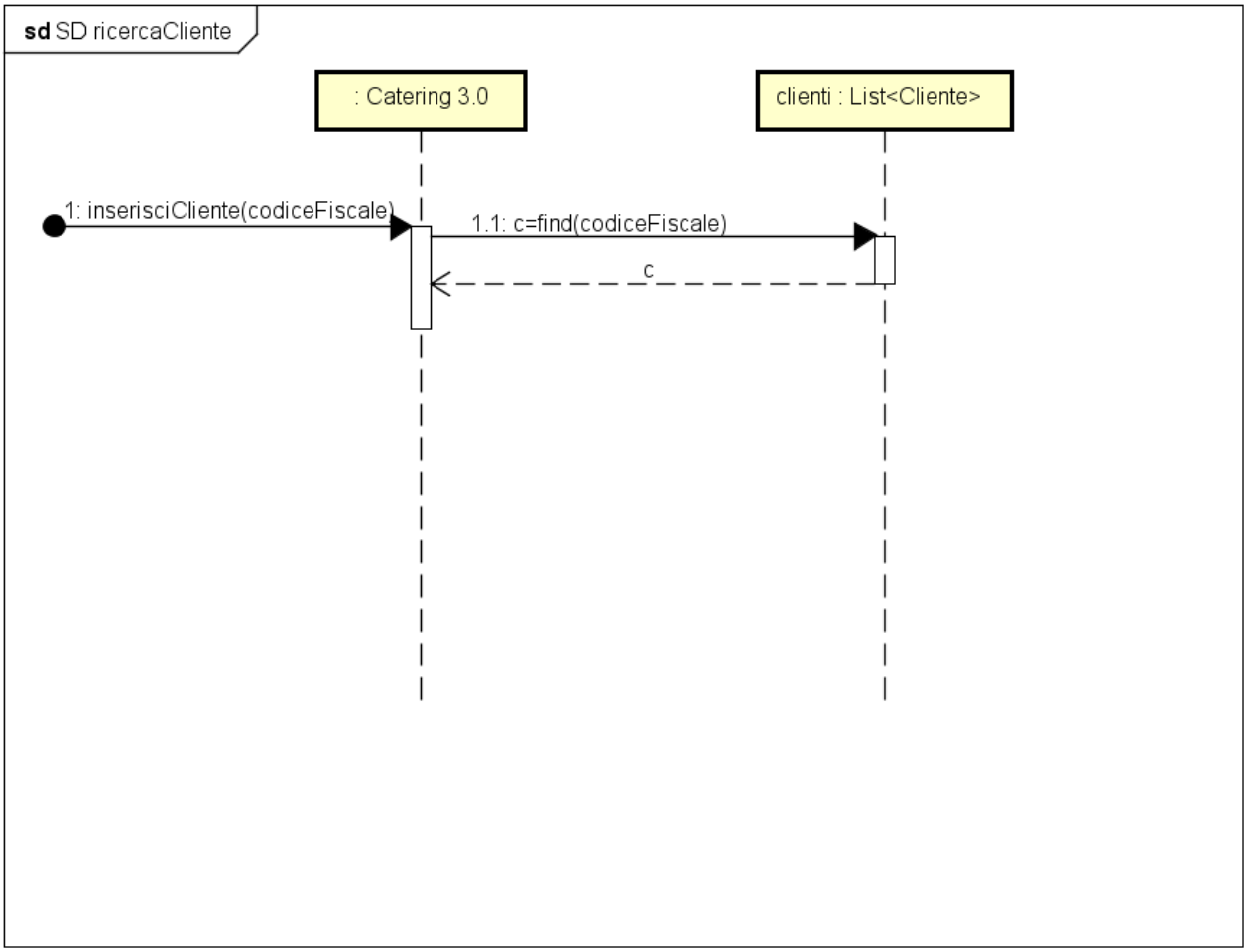
Operazione	findCateringbyRestaurant()
Riferimenti	Caso d'Uso UC7: Ricerca Caterings Ristorante
Pre-Condizioni	- è in corso la ricerca dei caterings di un ristorante
Post-Condizioni	- il Sistema visualizza i caterings del ristorante selezionato precedentemente.

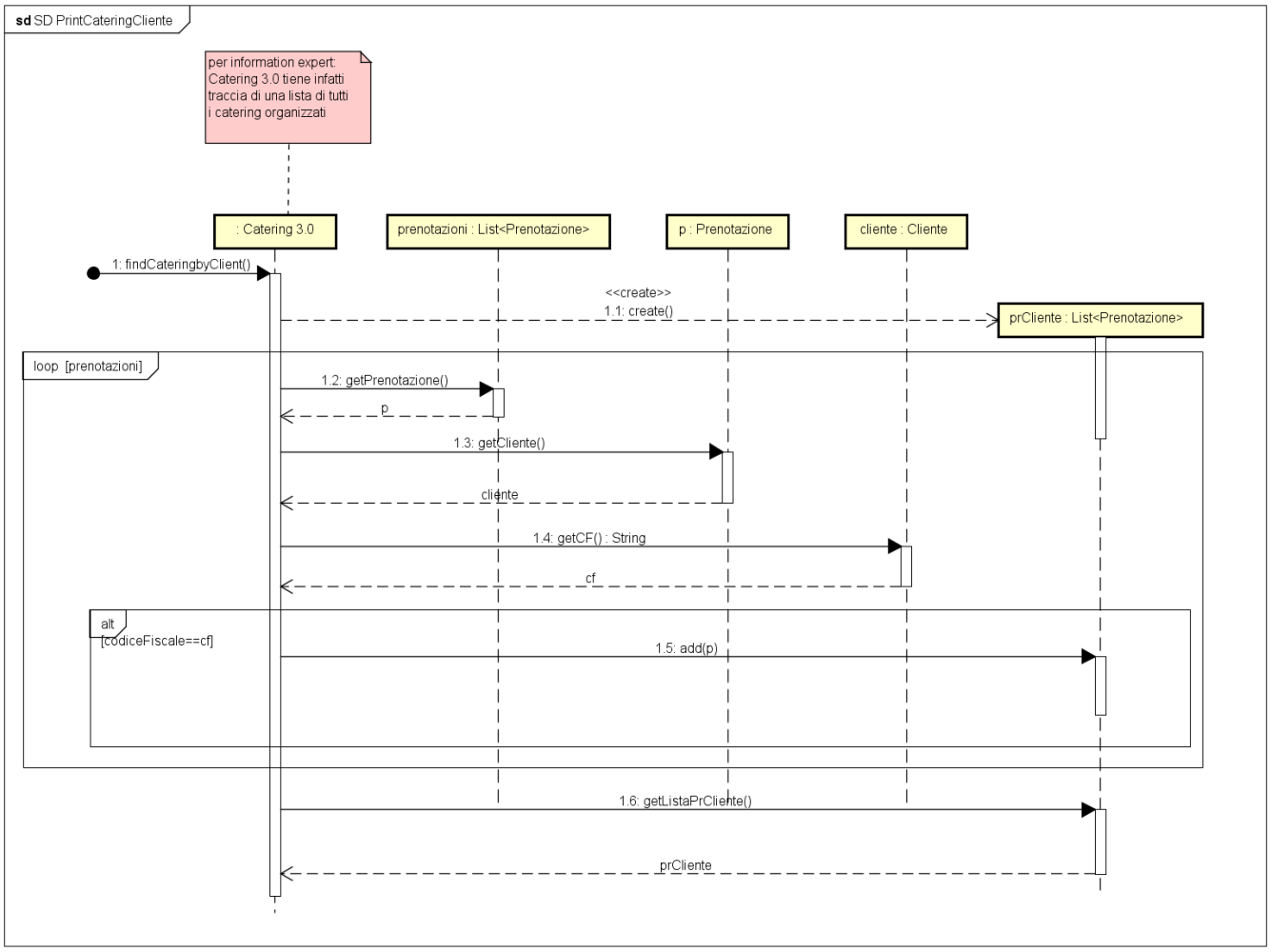
2.4 - Diagrammi di sequenza (SD)

Le operazioni trovate sono poi state dettagliate nei diagrammi di sequenza.

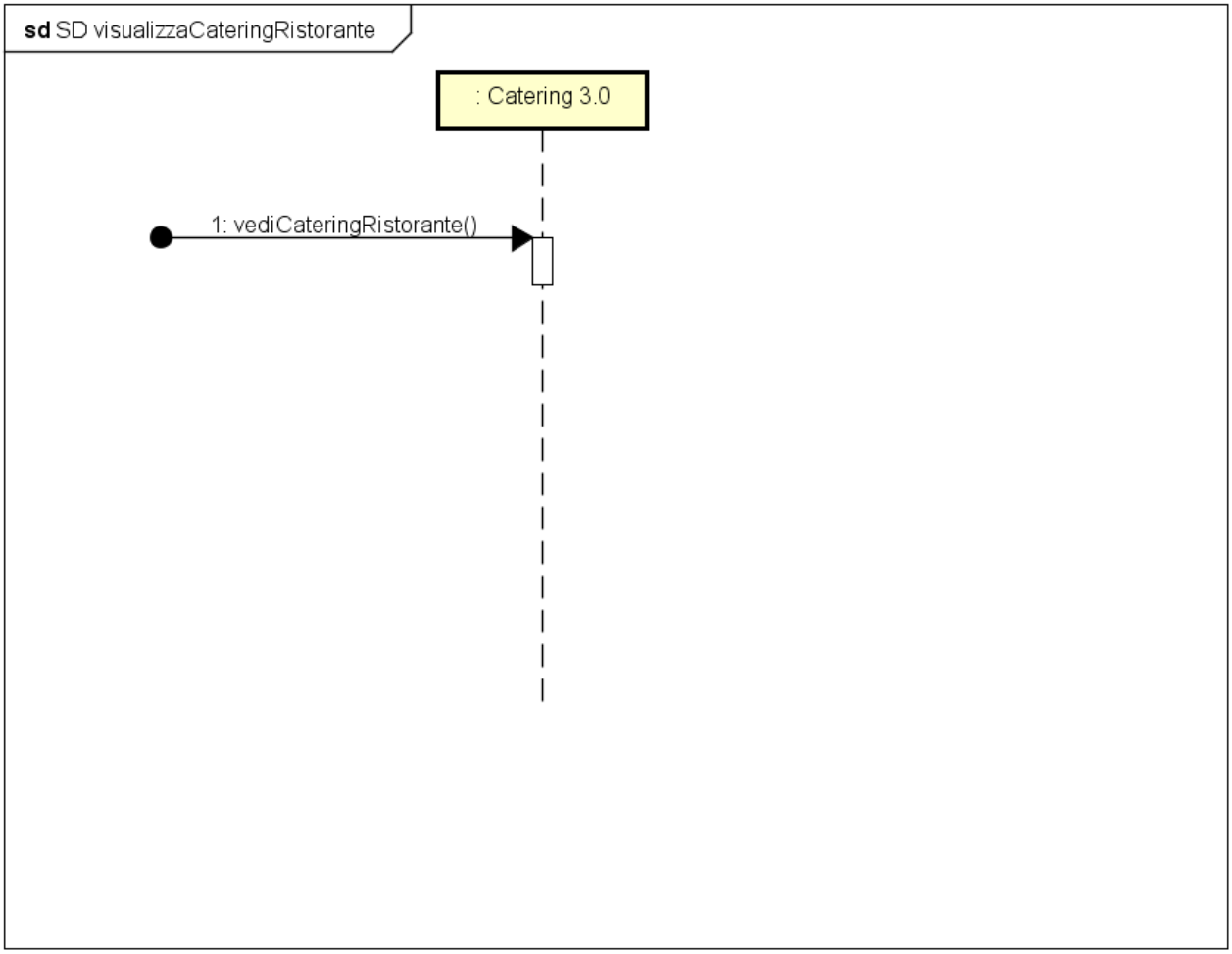
2.4.1 – SD caso d'Uso UC6

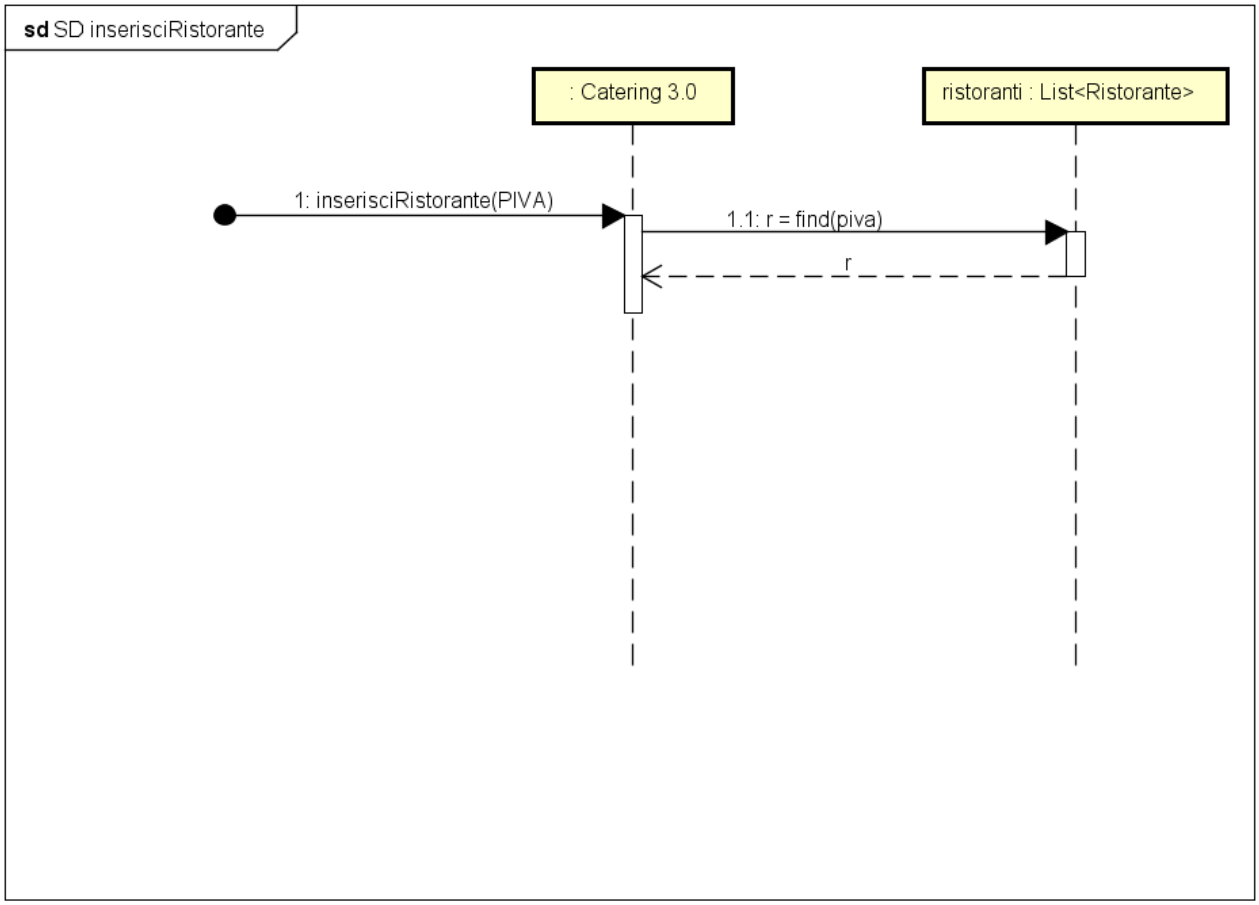


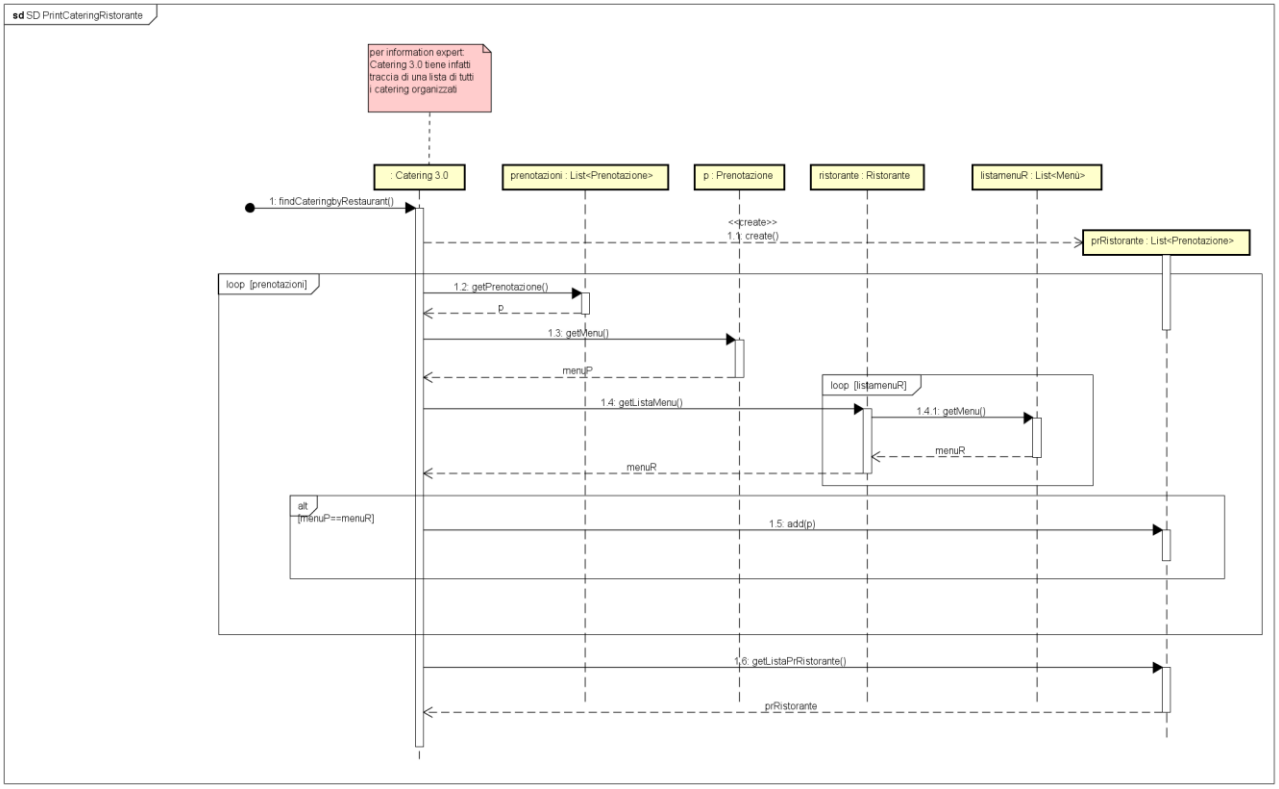

sd SD ricercaCliente
: Catering 3.0
clienti : List<Cliente>
1: inserisciCliente(codiceFiscale)
1.1: c=find(codiceFiscale)
c


sd SD PrintCateringCliente
per information expert:
Catering 3.0 tiene infatti
traccia di una lista di tutti
i catering organizzati
: Catering 3.0
prenotazioni : List<Prenotazione>
p : Prenotazione
cliente : Cliente
1: findCateringbyClient()
<<create>>
1.1: create()
prCliente : List<Prenotazione>
loop [prenotazioni]
1.2: getPrenotazione()
p
1.3: getCliente()
cliente
1.4: getCF() : String
cf
alt
[codiceFiscale==cf]
1.5: add(p)
1.6: getListaPrCliente()
prCliente

2.4.2 - SD Caso d’Uso UC7



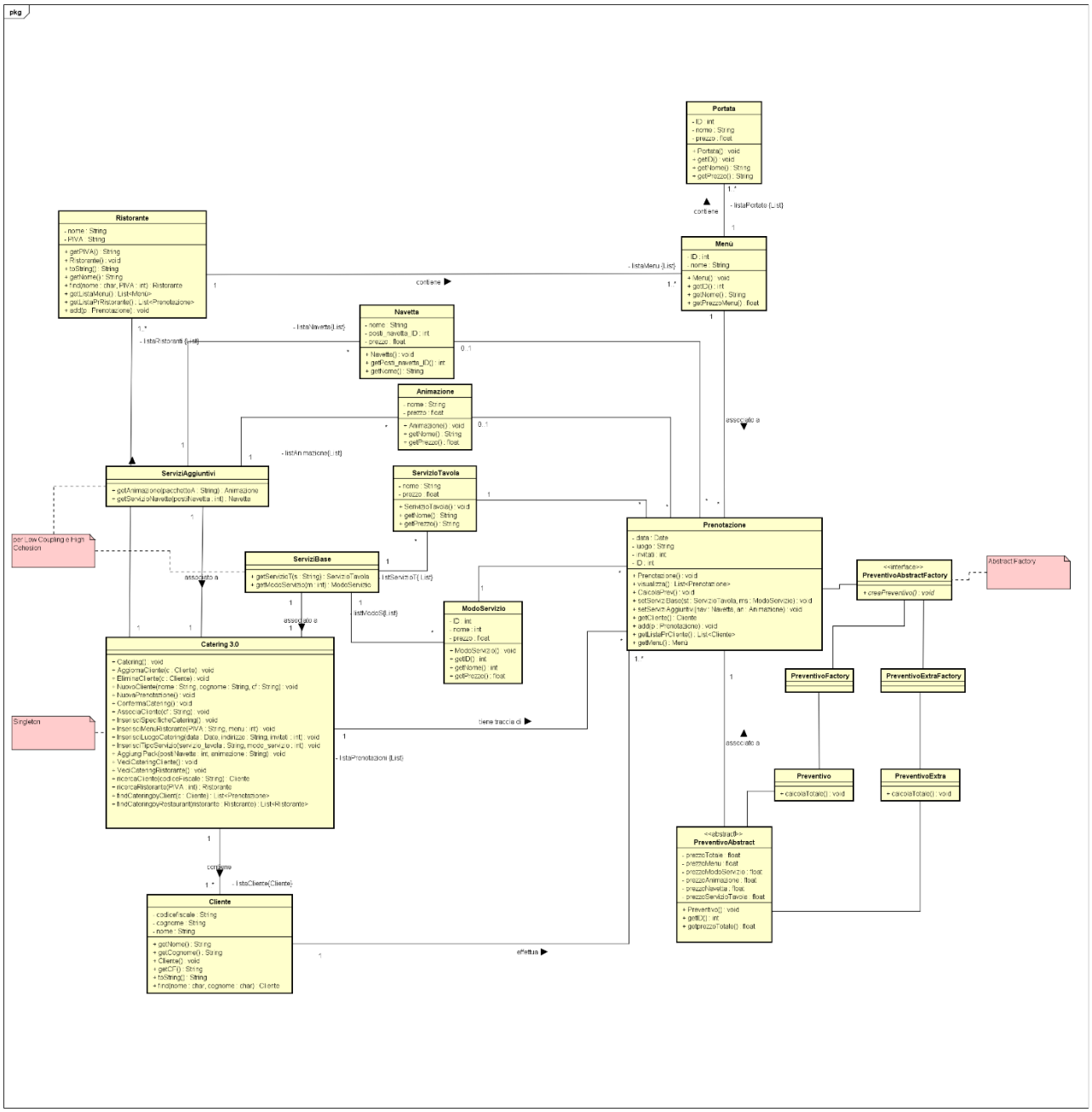

sd SD inserisciRistorante
: Catering 3.0
ristoranti : List<Ristorante>
1: inserisciRistorante(PIVA)
1.1: r = find(piva)
r


sd SD PrintCateringRistorante
per information expert:
Catering 3.0 tiene infatti
traccia di una lista di tutti
i catering organizzati
: Catering 3.0
prenotazioni : List<Prenotazione>
p : Prenotazione
ristorante : Ristorante
listamenuR : List<Menù>
1: findCateringbyRestaurant()
<<create>>
1.1: create()
prRistorante : List<Prenotazione>
loop [prenotazioni]
1.2: getPrenotazione()
p
1.3: getMenu()
menuP
loop [listamenuR]
1.4: getListaMenu()
1.4.1: getMenu()
menuR
menuR
alt
[menuP==menuR]
1.5: add(p)
1.6: getListaPrRistorante()
prRistorante

3 – Fase di Progettazione

3.1 – Diagrammi delle Classi di Progetto

Alla luce delle nuove funzionalità considerate e di quelle aggiornate, il Diagramma delle Classi di Progetto è stato modificato di conseguenza:



3.1.1 – Pattern applicati

- SINGLETON: è stato applicato il pattern creazionale singleton alla classe Catering 3.0, dal momento che questa rappresenta l intero sistema ed è quindi necessario che esista un' unica instanza di questa classe e che sia accessibile in maniera sicura e controllata.
- ABSTRACT FACTORY: è ora presente la classe PreventivoAbstractFactory, un' interfaccia da cui ereditano due classi factory dedicate, una per la versione semplice del preventivo, l' altra per una versione che richiede più informazioni.

4 – Codice Prodotto e Test

In fase di implementazione è stata creata una classe di testing usando JUnit. Sono state testate le funzionalità più importanti dell'applicativo, e la corretta inizializzazione delle variabili delle classi che vengono create durante i casi d'uso implementati. In particolare, sono stati raggruppati in tre casi di test i seguenti metodi:

- NuovoCliente(): il metodo esegue la creazione di un' istanza di cliente e l' inizializzazione dei suoi parametri, durante il test si è confrontato l' istanza creata col test con una creata appositamente verificando cosi il successo del metodo testato che ha passato tutti i test.
- RicercaCliente(): i dati di un Cliente inserito nel sistema e poi cercato tramite il metodo vengono confrontati con quelli di un'istanza creata appositamente uguale. Dall'esito positivo del test emerge che la ricerca viene effettuata correttamente.
- NuovoRistorante(): il metodo esegue la creazione di un' istanza di ristorante e l' inizializzazione dei suoi parametri, durante il test si è confrontato l' istanza creata col test con una creata appositamente verificando cosi il successo del metodo testato che ha passato tutti i test.
- RicercaRistorante(): i dati di un Ristorante inserito nel sistema e poi cercato tramite il metodo vengono confrontati con quelli di un'istanza creata appositamente uguale. Dall'esito positivo del test emerge che la ricerca viene effettuata correttamente
- InserimentoLuogoCatering(): vengono inseriti i dati luogo, data e invitati. Nel test vengono confrontati i valori dati in ingresso con quelli restituiti dai getter e i setter propri della classe. Tutti i test sono stati passati dal metodo che si dimostra quindi corretto.
- InserisciTipoServizio(): vengono inseriti i dati tipo servizio e servizio tavola. Nel test vengono confrontati i valori dati in ingresso con quelli restituiti dai getter e i setter propri della classe. Tutti i test sono stati passati dal metodo che si dimostra quindi corretto.
- AggiungiPack(): vengono inseriti i dati navetta e animazione. Nel test vengono confrontati i valori dati in ingresso con quelli restituiti dai getter e i setter propri della classe. Tutti i test sono stati passati dal metodo che si dimostra quindi corretto.
- AssociaCliente(): il metodo setta il campo cliente nella classe prenotazione in maniera da avere un' associazione diretta fra la prenotazione in questione e il cliente che la effettua. Il test si è occupato di verificare che il parametro settato sia uguale a quello dato in ingresso e quindi vedere il funzionamento del setter che è risultato corretto.
- RicercaCateringsCliente(): il seguente metodo si occupa del recupero delle istanze di catering associate ad un cliente. Nel test è stato creato un catering e associato ad un cliente

in seguito si è chiamato il metodo di ricerca e sono stati confrontati i risultati della ricerca con quelli inseriti, il test ha avuto esito positivo.

- RicercaCateringsRistorante(): il seguente metodo si occupa del recupero delle istanze di catering associate ad un ristorante. Nel test è stato creato un catering e associato ad un ristorante in seguito si è chiamato il metodo di ricerca e sono stati confrontati i risultati della ricerca con quelli inseriti, il test ha avuto esito positivo.