2.1 Iterazione 2, Analisi

2.1.1 Introduzione

Per l'iterazione 2 sono stati scelti i seguenti requisiti:

- lo scenario principale di successo del caso d'uso UC1 (Nuovo ordine al fornitore);
- le estensioni 1a, 1b e 1c del caso d'uso UC2;
- caso d'uso d'avviamento.

2.1.2 Caso d'uso UC1: Nuovo ordine al fornitore

Scenario principale

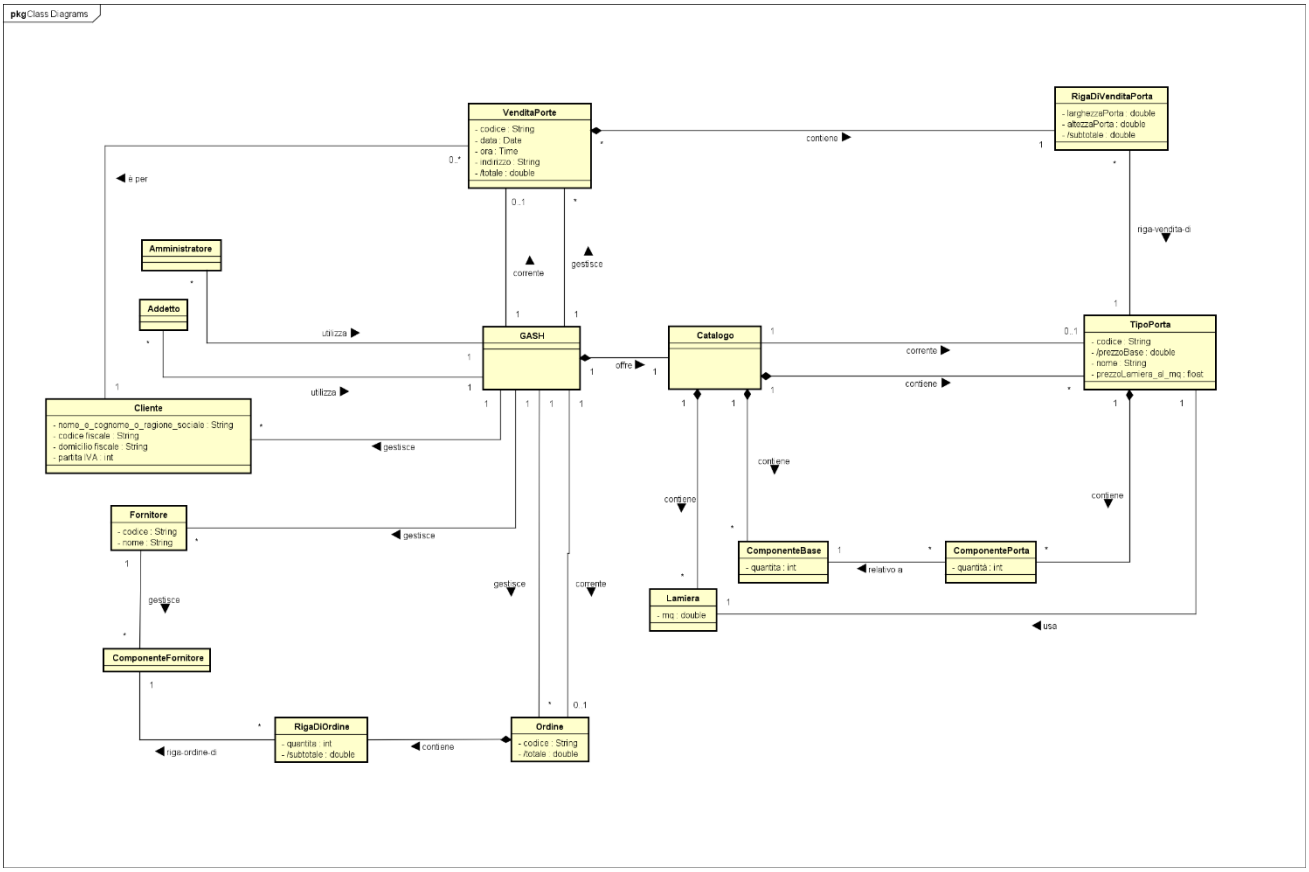
1. L'Amministratore vuole ordinare dei componenti.
2. L'Amministratore sceglie la voce nel programma "Nuovo ordine MO".
3. L'Amministratore inserisce il codice del Fornitore. Il Sistema mostra i componenti che si ordinano abitualmente dal Fornitore.
4. L'Amministratore inserisce il codice del componente da ordinare e digita la quantità. Il Sistema registra il componente richiesto e la relativa quantità.

   Il passo 4 viene ripetuto finché serve.
5. L'Amministratore indica di aver finito. Il Sistema calcola il prezzo dei componenti ordinati e mostra il riepilogo delle informazioni sull'ordine.
6. L'Amministratore conferma l'ordine. Il Sistema registra le informazioni relative all'ordine. Il Sistema genera un Modulo d'ordine stampabile (in pdf).
7. L'Amministratore invia tramite email il Modulo d'ordine.

In quest'iterazione del caso d'uso UC1 viene analizzato interamente lo scenario principale di successo. Da esso possono essere ricavate le seguenti classi, non ancora identificate nei casi d'uso precedenti:

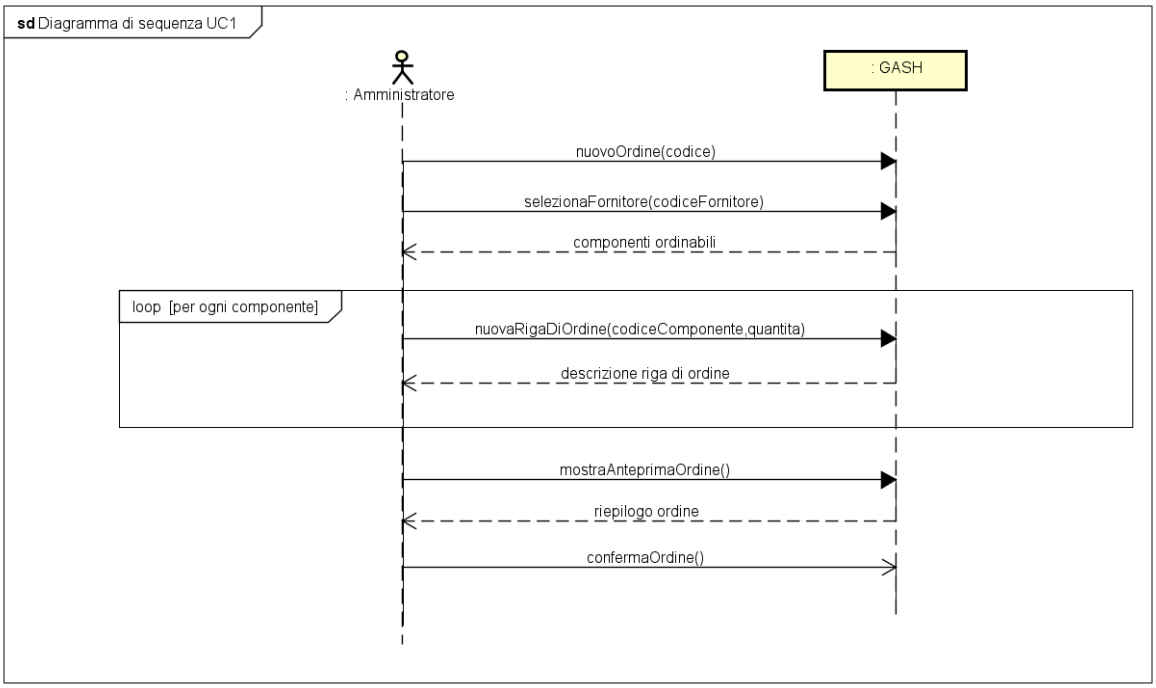
- **Fornitore**: un fornitore di componenti ordinabili.
- **Ordine**: un ordine effettuato dall'Amministratore, composto da uno o più componenti.
- **ComponenteFornitore**: un componente ordinabile, mantenuto da un fornitore.
- **RigaDiOrdine**: contiene le informazioni riguardanti l'ordine al fornitore di un elemento, corrispondente ad un componente.

Modello di dominio



2.1.3 Caso d'uso UC1, Diagramma di sequenza di sistema

Il diagramma di sequenza di sistema per il caso d'uso UC1 è il seguente:



2.1.4 Caso d'uso UC1, Contratti delle operazioni

2.1.4.1 Nuova riga di ordine

L'operazione di sistema nuovaRigaDiOrdine consente di aggiungere un nuovo componente all'ordine corrente.

Operazione	nuovaRigaDiOrdine(codiceComponente: String, quantita: int)
Riferimenti	Caso d'uso: *Nuovo ordine al fornitore*
Pre-condizioni	- è in corso l'inserimento di un ordine o
Post-condizioni	- è stata creata una nuova istanza *ro* di RigaDiOrdine; - la riga di ordine *ro* è stata associata a un ComponenteFornitore *cf*, sulla base di codiceComponente, tramite l'associazione "registra-ordine-di"; - gli attributi di *ro* sono stati inizializzati (l'attributo /subtotale viene calcolato come segue: subtotale = prezzoComponente x quantita); - la riga di ordine *ro* è stata associata all'Ordine corrente tramite l'associazione "contiene".

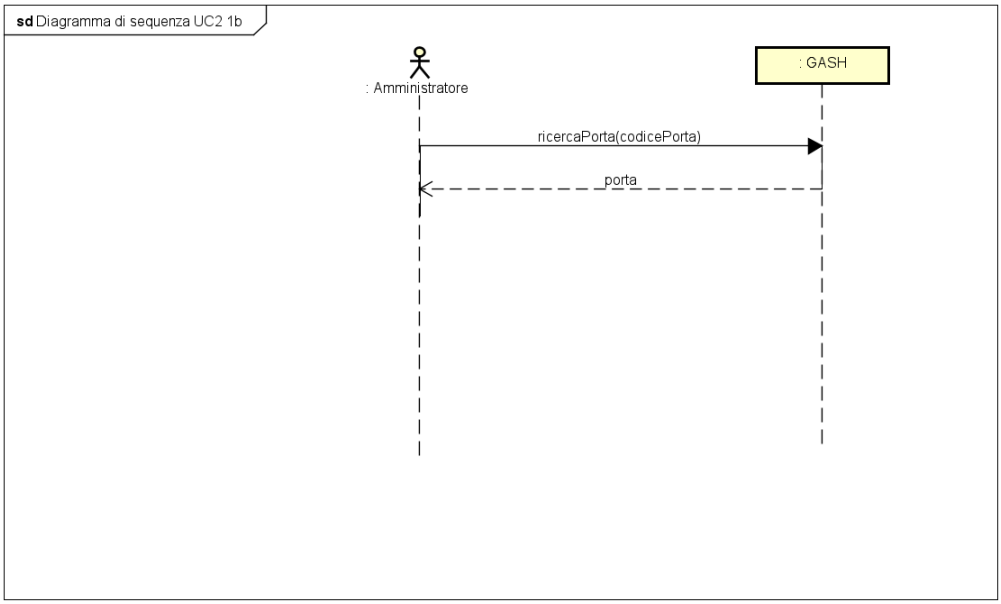
2.1.5 Caso d'uso UC2: Inserimento nuovo tipo di Porta per garage (scenari alternativi), Modello di dominio

In quest'iterazione del caso d'uso UC2 vengono analizzate le estensioni 1a, 1b e 1c. Lo sviluppo di questi scenari alternativi non comporta l'aggiunta di nuove classi al modello di dominio.

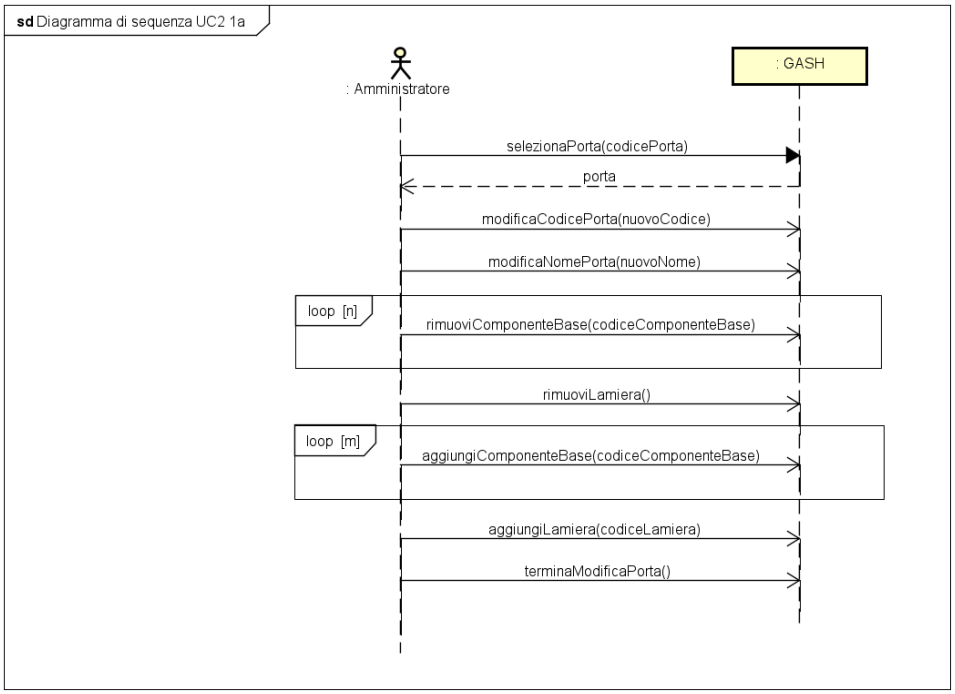
2.1.6 Caso d'uso UC2 (scenari alternativi), Diagramma di sequenza di sistema

l diagrammi di sequenza di sistema che descrivono gli scenari alternativi del caso d'uso UC2 sono i seguenti:

Estensione 1a (ricerca)

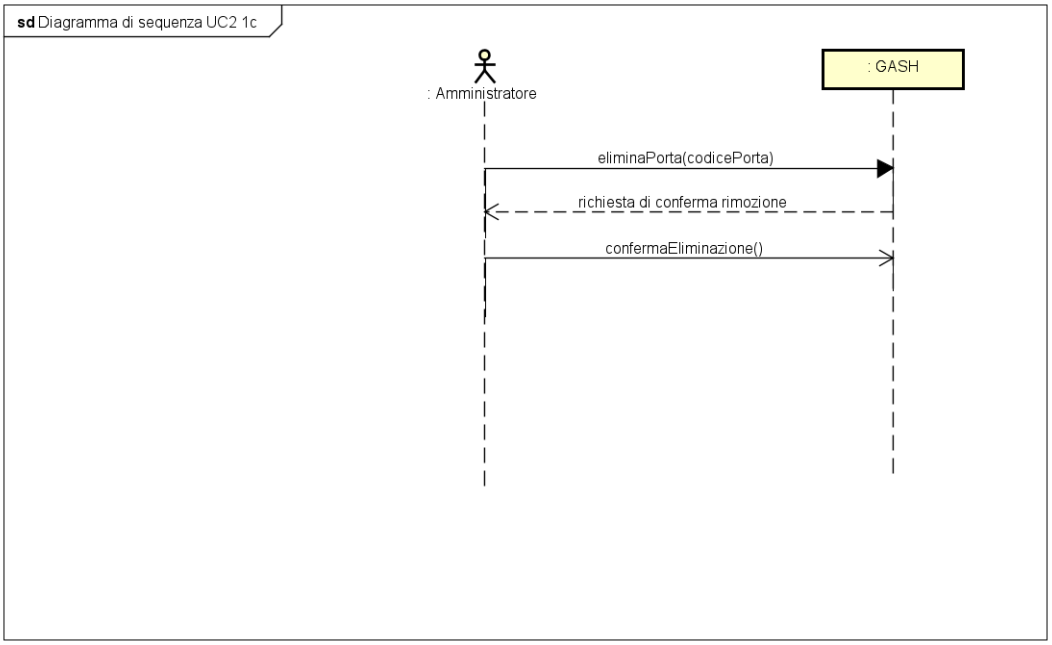


Estensione 1b (modifica)



Durante la fase di progettazione, verrà implementato un selettore che ha come opzioni le diverse operazioni di modifica.

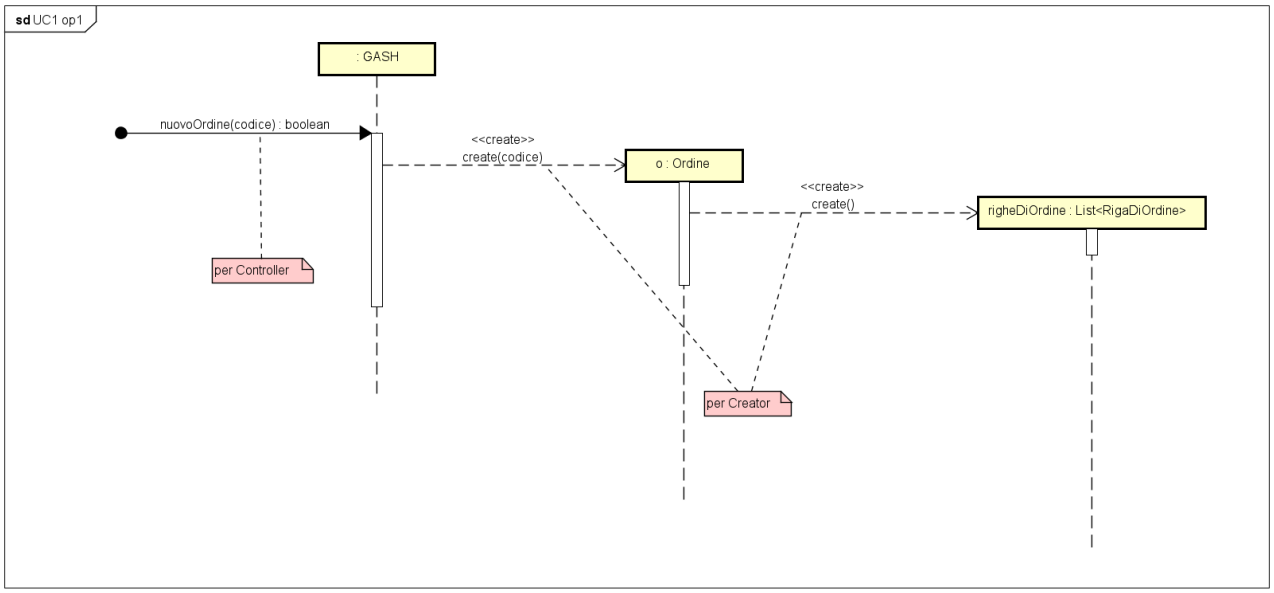
Estensione 1c (eliminazione)



2.2 Iterazione 2, Progettazione

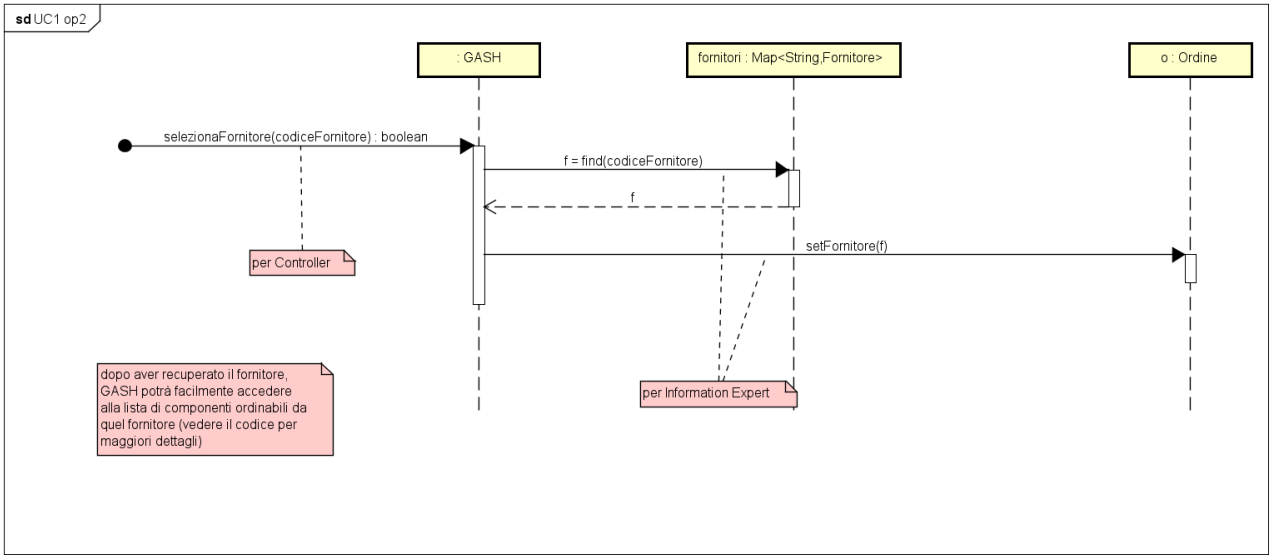
2.2.1 Caso d'uso UC1, Diagrammi di interazione

2.2.1.1 nuovoOrdine(codice:String)

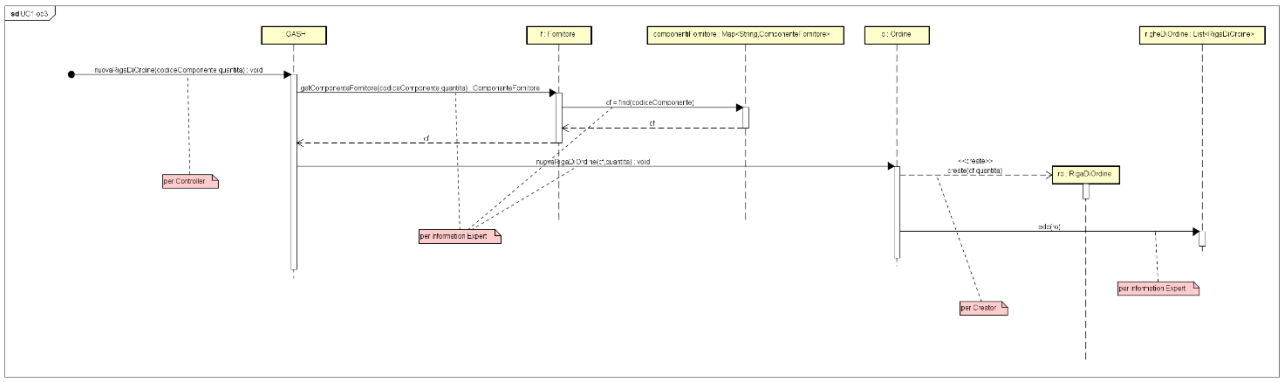


L'operazione nuovoOrdine ritorna una variabile Booleana: se l'oggetto di Ordine *o* viene istanziato correttamente ritorna *true*, altrimenti *false*.
Durante la fase di implementazione verranno effettuati dei controlli sul codice inserito, ad esempio un controllo sull'esistenza di un ordine con quel codice nel sistema e un controllo sul formato del codice (inserimento di una stringa vuota).

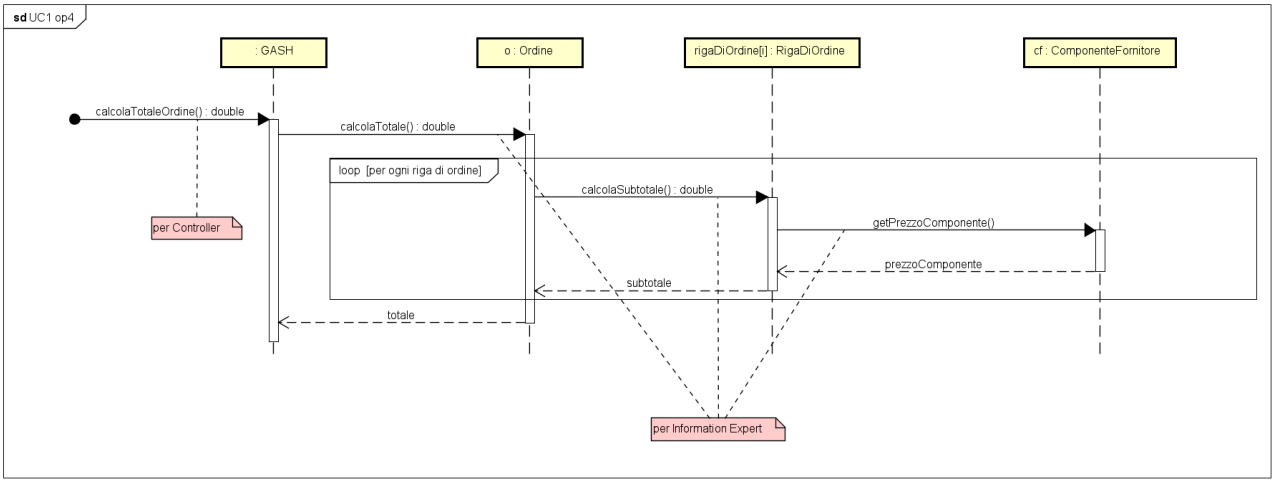
2.2.1.2 selezionaFornitore(codiceFornitore:String)



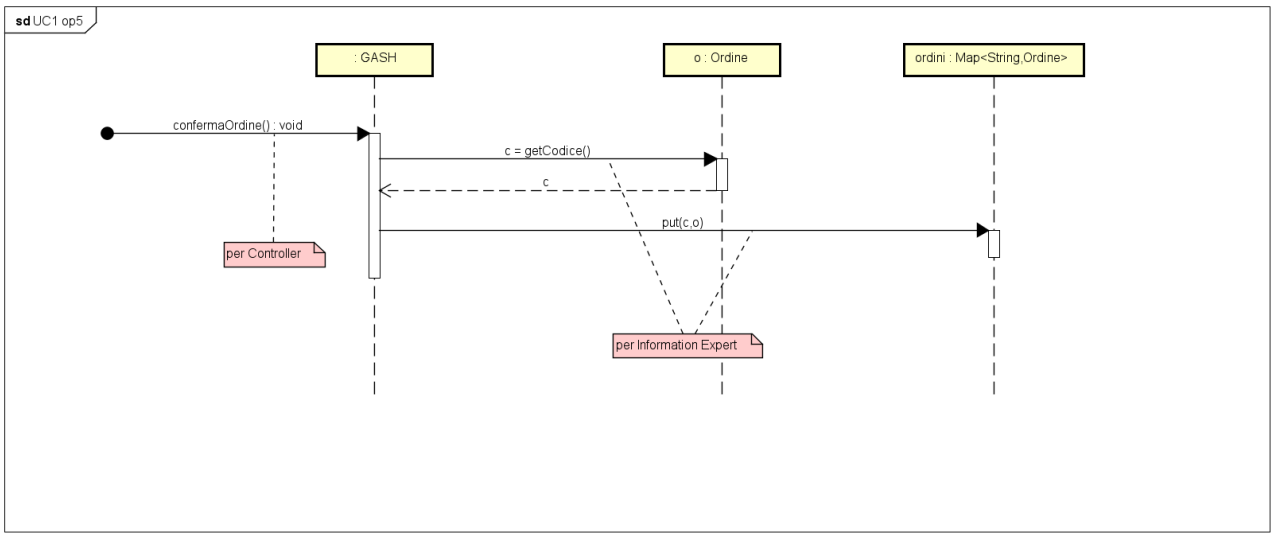
2.2.1.3 nuovaRigaDiOrdine(codiceComponente:String, quantita:int)



2.2.1.4 calcolaTotaleOrdine()



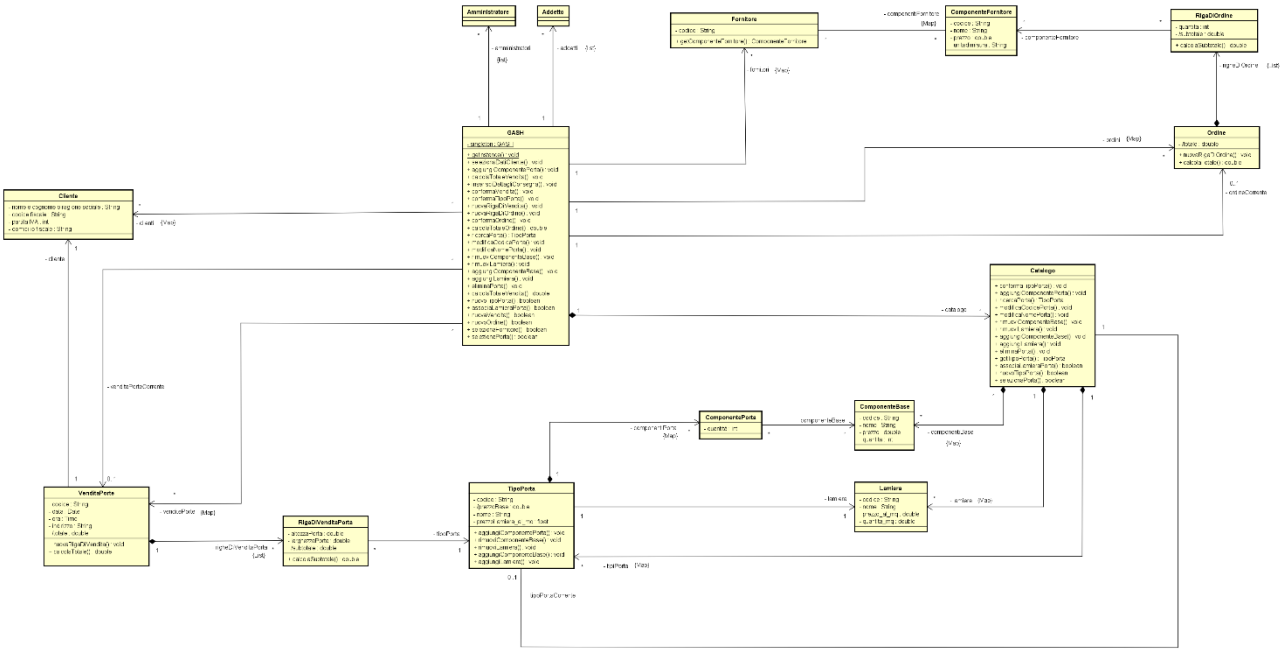
2.2.1.5 confermaOrdine()



Il metodo setDataEOraVendita() imposterà la data e ora attuale dell'ordine mediante una funzione di una libreria java (`LocalDateTime`).

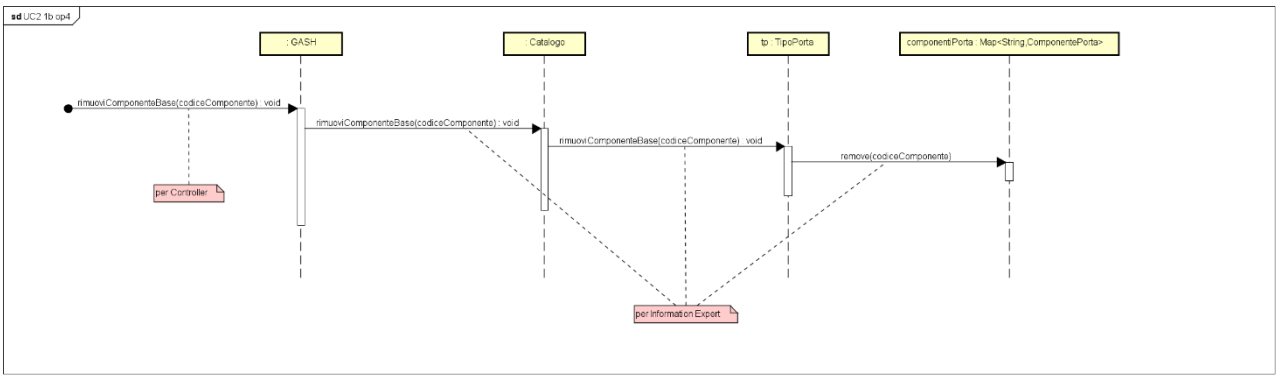
Tutti gli ordini verranno registrati in un file di testo.

2.2.2 Caso d'uso UC1, Diagramma delle classi di progetto



2.2.3 Caso d'uso UC2 (estensione 1a - ricerca), Diagrammi di interazione

2.2.3.1 ricercaPorta(codicePorta:String)

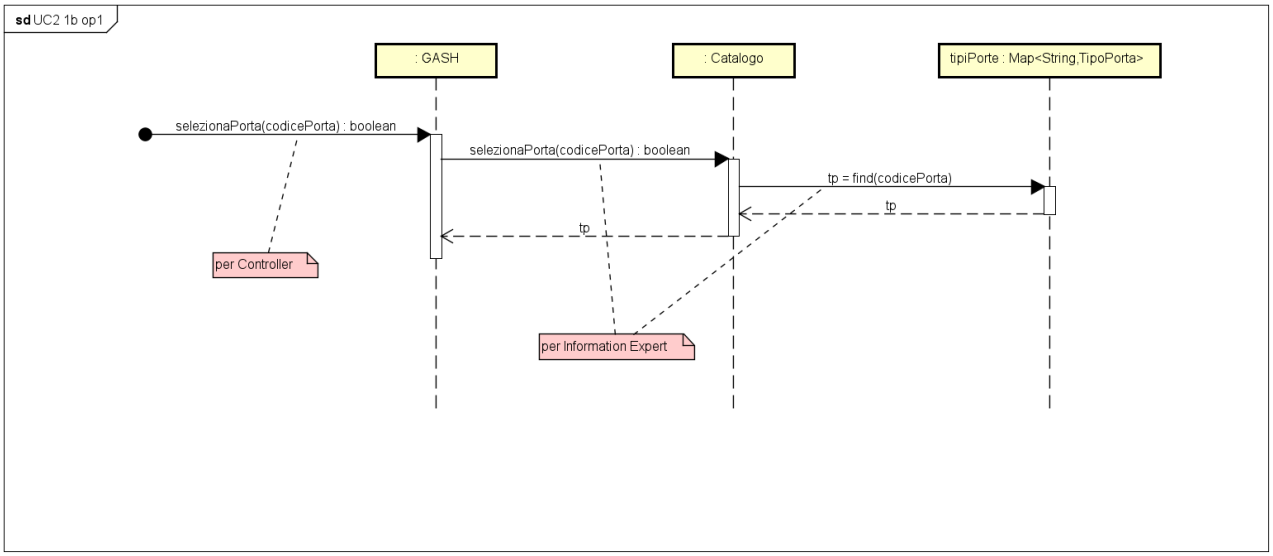


2.2.4 Caso d'uso UC2 (estensione 1b - modifica), Diagrammi di interazione

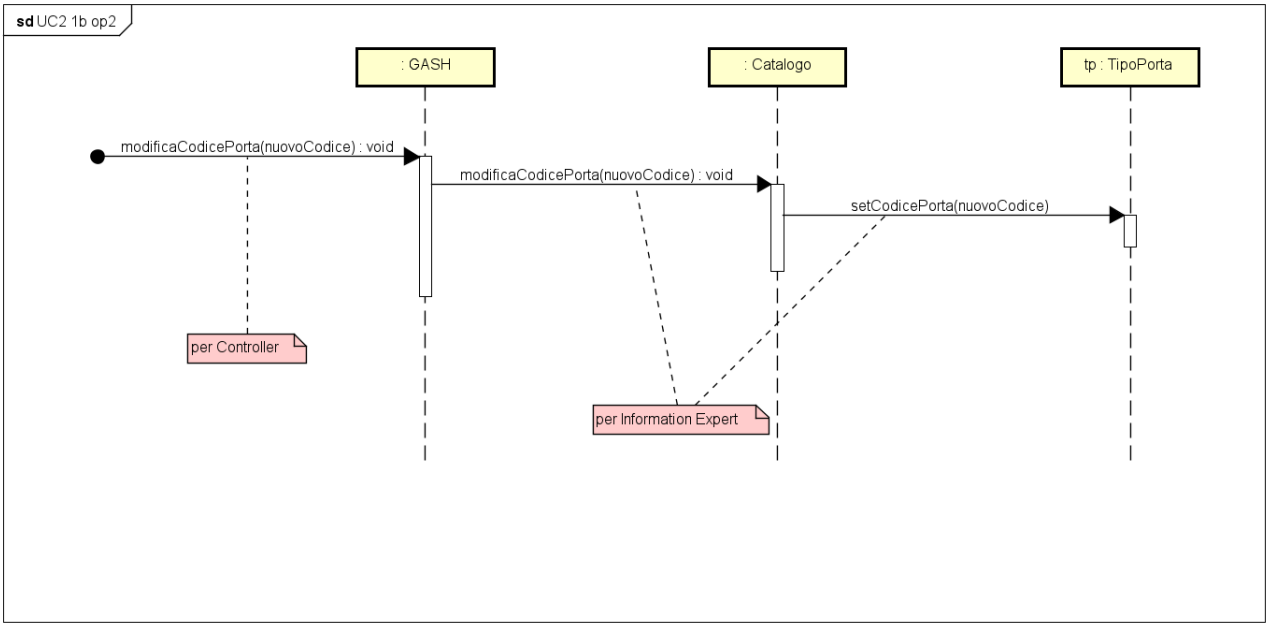
Quando l'Amministratore eseguirà questo caso d'uso, potrà selezionare una tra le opzioni disponibili, che corrispondono alle diverse operazioni di questo diagramma (modifica del codice, del nome, rimozione o aggiunta di componenti, ecc.).

Questa scelta sarà possibile grazie all'implementazione di un opportuno selettore nella GUI, che comparirà subito dopo l'esecuzione dell'operazione selezionaPorta().

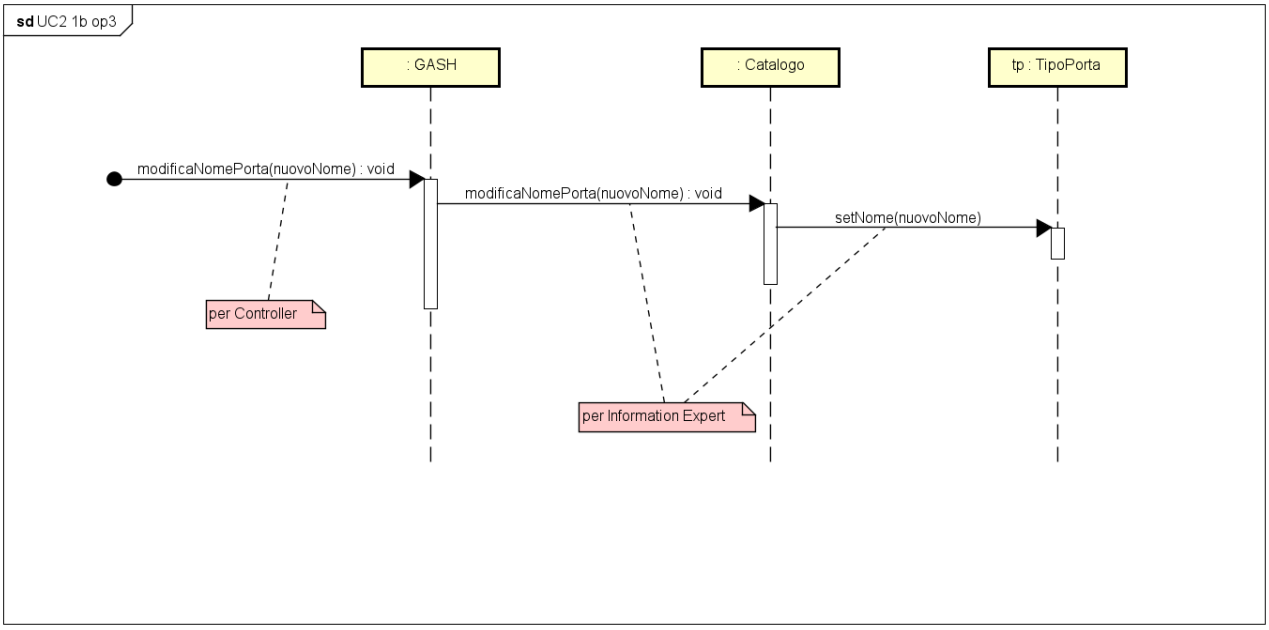
2.2.4.1 selezionaPorta(codicePorta:String)



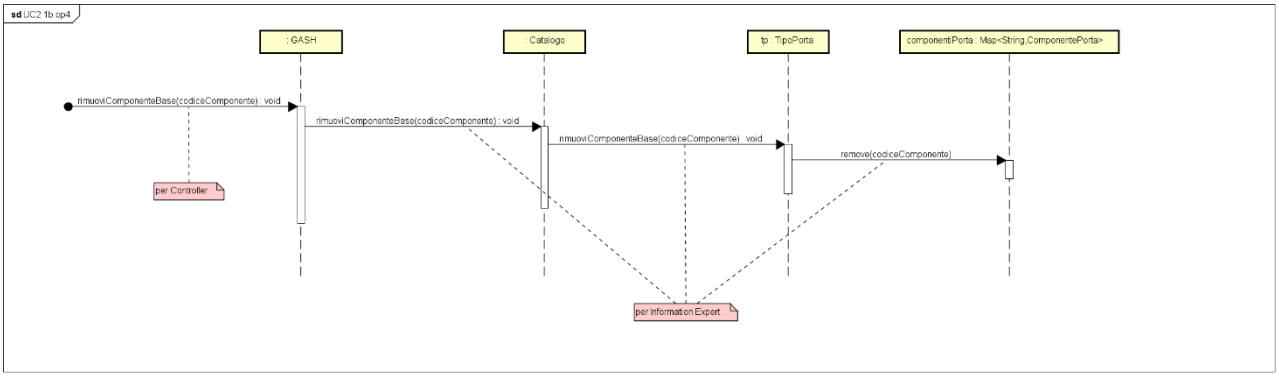
2.2.4.2 modificaCodicePorta(nuovoCodice:String)



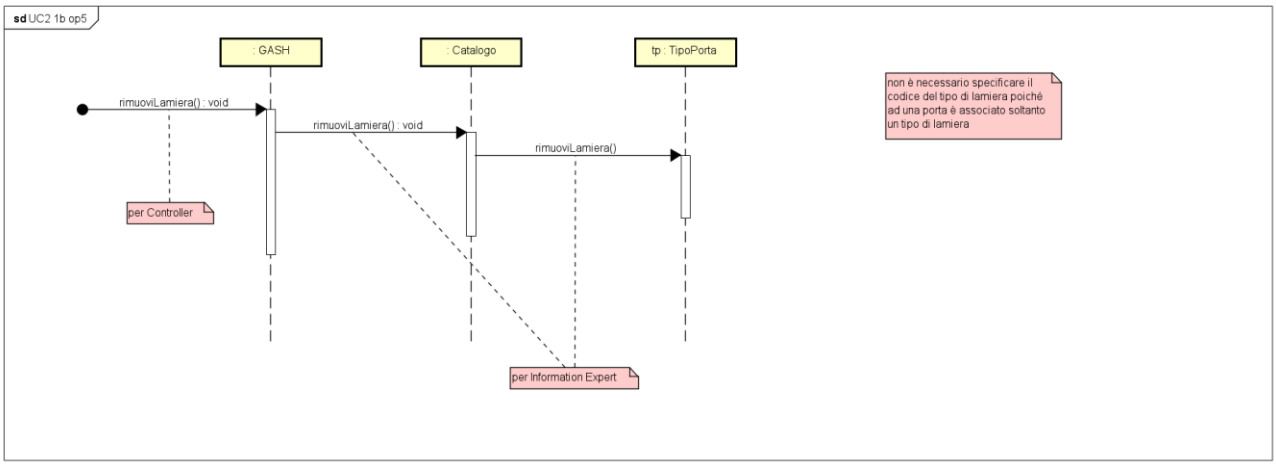
2.2.4.3 modificaNomePorta(nuovoNome:String)



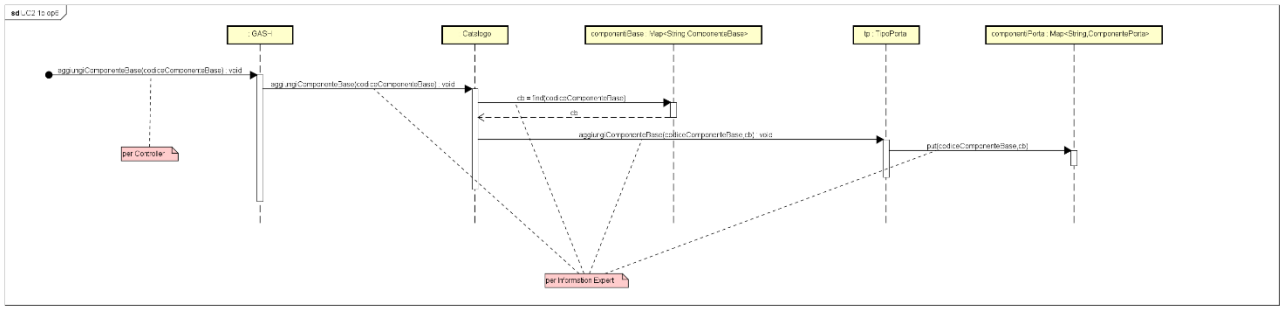
2.2.4.4 rimuoviComponenteBase(codiceComponente:String)



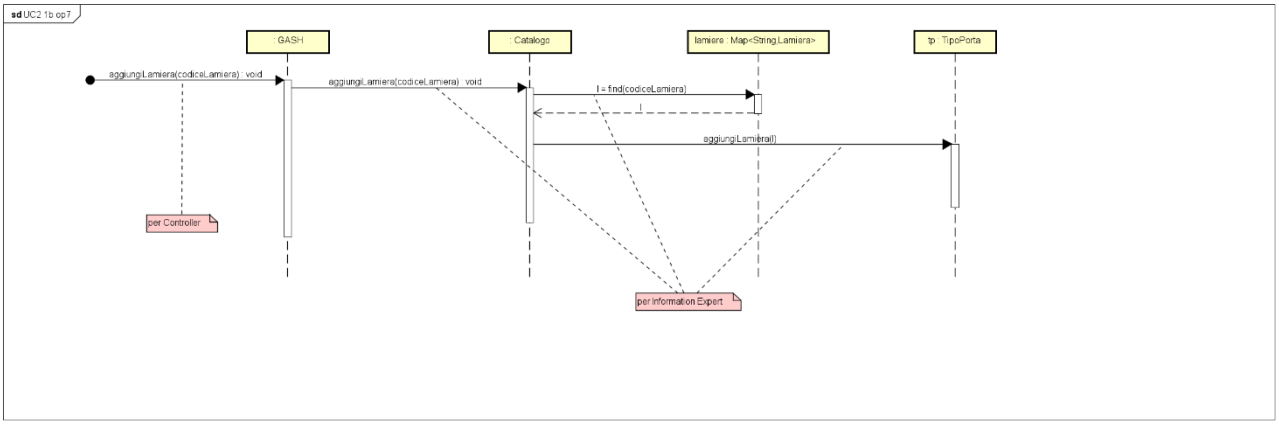
2.2.4.5 rimuoviLamiera()



2.2.4.6 aggiungiComponenteBase(codiceComponenteBase:String)

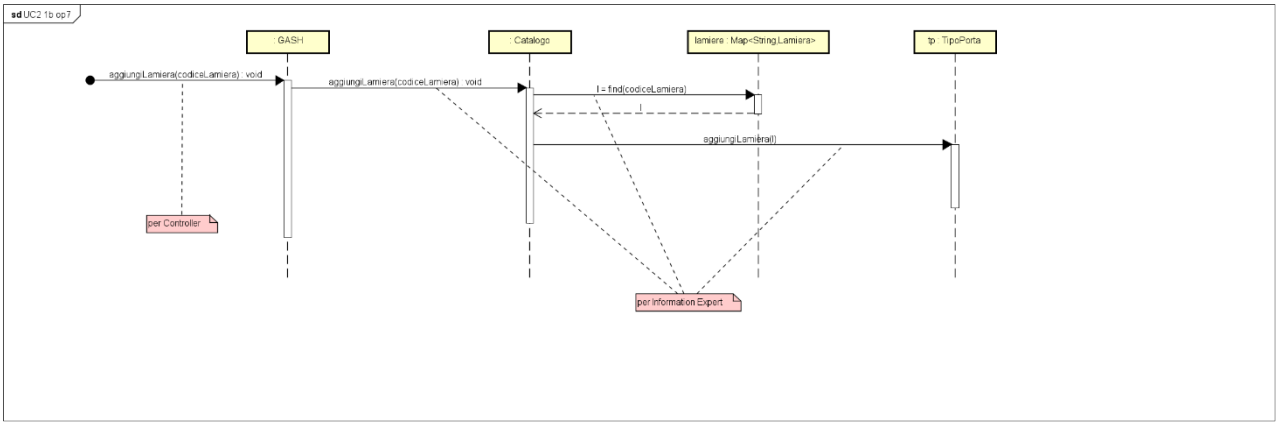


2.2.4.7 aggiungiLamiera(codiceLamiera:String)



2.2.5 Caso d'uso UC2 (estensione 1c - eliminazione), Diagrammi di interazione

2.2.5.1 eliminaPorta(codicePorta:String)



2.3 Iterazione 2, Refactoring

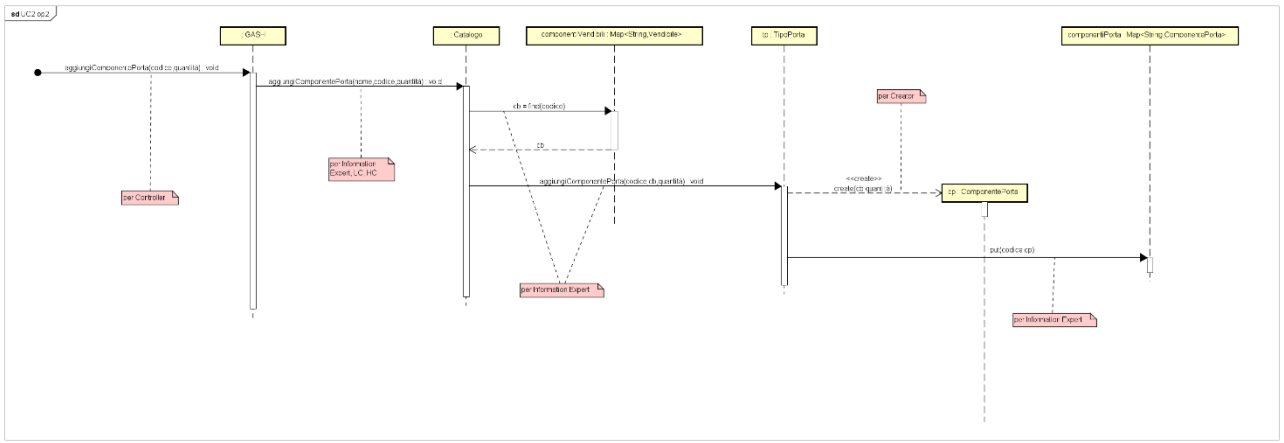
2.3.1 Introduzione

Analizzando il codice del progetto, si nota che le classi ComponenteBase e Lamiera sono strettamente correlate (in quanto entrambe rappresentano un tipo di componente) e condividono la maggior parte degli attributi. Esse, inoltre, hanno proprietà strutturali aggiuntive di interesse. Risulta, quindi, utile introdurre una classe Componente, da cui ComponenteBase e Lamiera ereditano il comportamento (polimorfismo). Questa modifica semplificherà il codice e ne favorirà il riuso, anche nel caso in cui dovranno essere aggiunti in futuro nuovi tipi di componenti, caratterizzati da attributi e associazioni aggiuntive. Nella classe Componente va aggiunto un nuovo attributo, unità di misura, che identificherà la tipologia di prezzo che caratterizza un componente (ad esempio prezzo unitario, prezzo al mq, ecc.).
Inoltre, si nota che anche la classe ComponenteFornitore condivide gli stessi attributi di ComponenteBase e Lamiera. Anche in questo caso, quindi, è conveniente far sì che ComponenteFornitore sia una sottoclasse di Componente. L'unica differenza fra ComponenteFornitore e le due sottoclassi di Componente appena analizzate riguarda l'attributo quantità, che in ComponenteFornitore non è presente, e l'attributo prezzo, in quanto il prezzo di vendita di un componente sarà sicuramente diverso (più nello specifico, maggiore) del prezzo di acquisto. Per il risolvere il primo problema è sufficiente inserire un attributo quantità nella sottoclasse ComponenteBase e un attributo mq (metri quadri) nella sottoclasse Lamiera. Invece, per risolvere il secondo problema, è stato inserito un attributo prezzo nella superclasse Componente ed è stata introdotta una nuova interfaccia, Vendibile, implementata da ComponenteBase e Lamiera. Tale interfaccia è caratterizzata da un metodo getPrezzo(), che permetterà di modificare il prezzo (di acquisto) di un componente con il nuovo prezzo di vendita.
Infine, per tenere traccia dello stato di un ordine, è stata creata una nuova classe StatoOrdine, collegata alla classe Ordine tramite l'associazione "si-trova-in", a cui sono associati gli stati "InArrivo" e "Processato" (pattern State). Non appena viene effettuato un nuovo ordine, il suo stato viene impostato in "InArrivo". Al termine della registrazione dei componenti nel sistema lo stato diventerà "Processato".

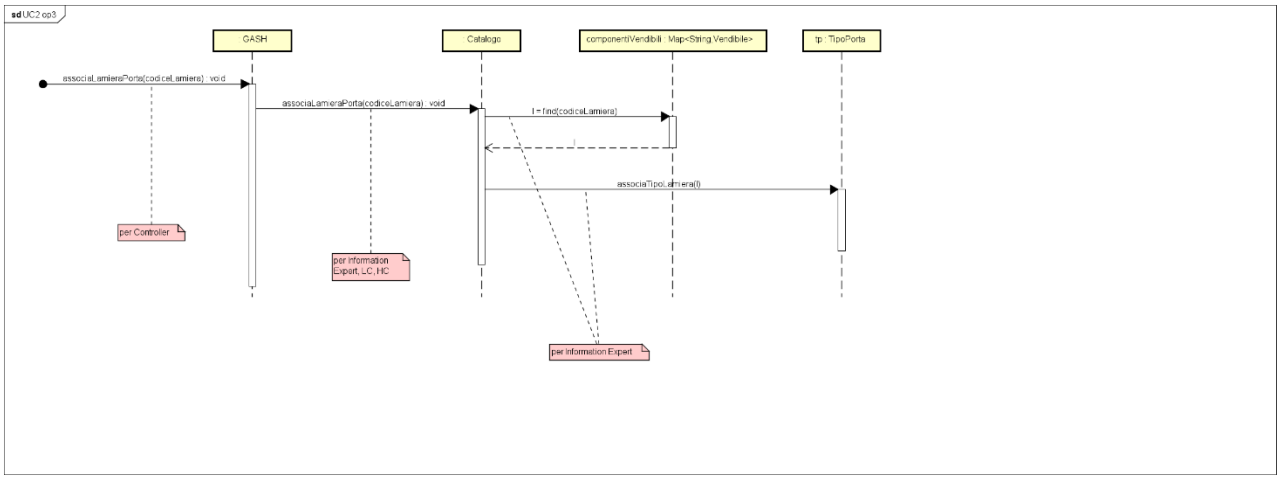
2.3.2 Caso d'uso UC2

Le operazioni di refactoring applicate (in particolare l'introduzione della classe Componente) comportano la modifica di alcuni diagrammi di interazione relativi al caso d'uso UC2.

2.3.2.1 aggiungiComponentePorta(codice:String,quantita:int):void

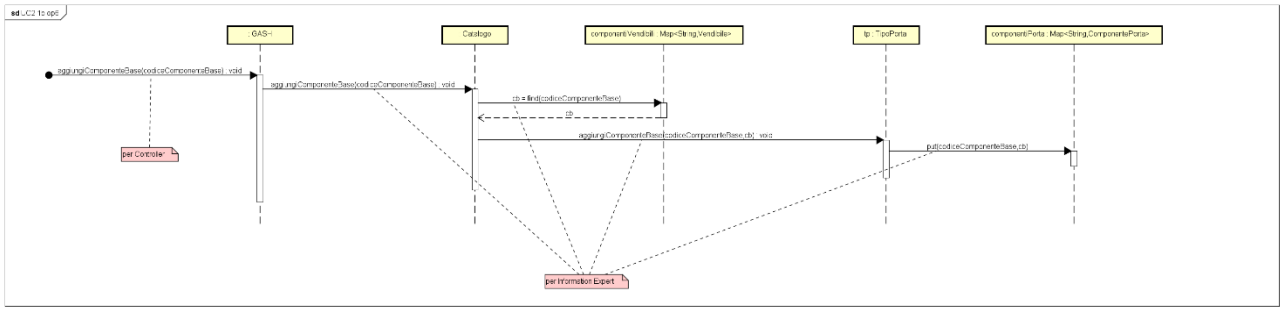


2.3.2.2 associaLamieraPorta(codiceLamiera:String):void

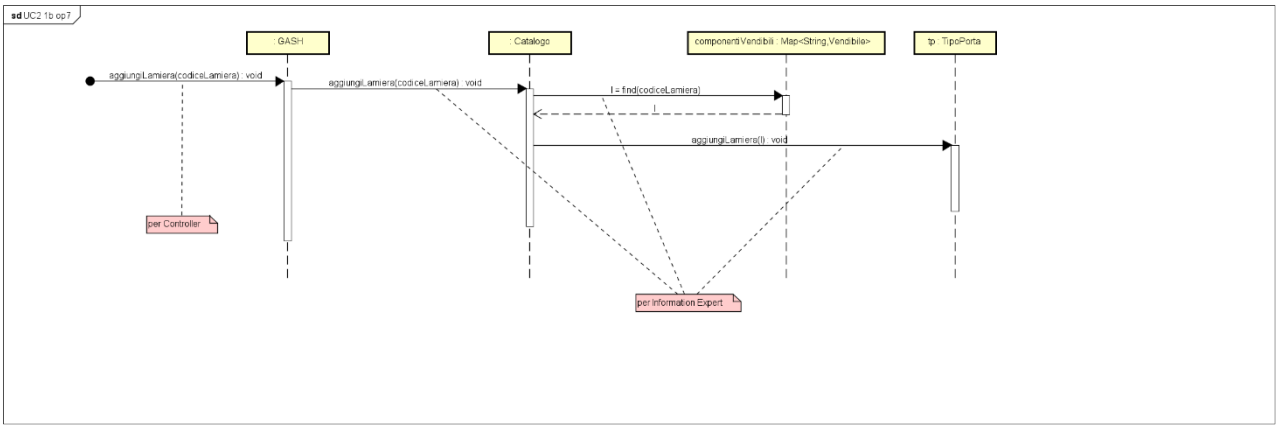


2.3.3 Caso d'uso UC2 (estensione 1b)

2.3.3.1 aggiungiComponenteBase(codiceComponenteBase:String):void



2.3.3.2 aggiungiLamiera(codiceLamiera:String):void



2.3.4 Diagramma delle classi di progetto