

# Room8

---

Il software per condividere le tue spese

# Descrizione del problema

**Room8** è per chiunque abbia bisogno di dividere tra più persone **conti di qualunque tipo**. Grazie a questo software, infatti, tutto quello che è necessario fare è creare un gruppo, e aggiungere volta volta le spese effettuate, specificando chi si è occupato di pagare sul momento. Alla fine si saprà esattamente quanto è necessario che ognuno renda per **pareggiare i conti**.

Grazie alla **gestione simultanea di vari gruppi** è possibile utilizzarlo sia per una situazione stabile (ad esempio con i propri **compagni di stanza**, o col proprio **partner**), sia per un evento occasionale come un **viaggio**, nel quale sia necessario dividere le spese di rifornimenti, biglietti, cibo...



E' richiesto lo sviluppo di un software per la gestione di **movimenti di denaro** tra due o più utenti, facenti parte di uno stesso **gruppo**.

Di ogni **utente** occorre conoscere: l'indirizzo e-mail, la password di accesso, la denominazione (cognome e nome), il telefono e la foto (opzionale).

Di ogni **gruppo** occorre conoscere: il nome, la lista dei membri e la foto (opzionale).

Le foto, se assenti, vengono impostate con un'immagine di default. Uno stesso utente può far parte di più gruppi e tutti gli utenti che appartenenti ad uno stesso gruppo diventano **amici**.





Ogni qualvolta un utente intende inserire una **spesa**, deve specificare, il gruppo coinvolto, una descrizione, l'importo totale speso, l'utente pagante, il **metodo di divisione** e la data.

In base al **metodo di divisione** scelto l'importo della spesa viene suddiviso tra i vari utenti coinvolti, in modo da definire chiaramente la somma sorta a credito, o a debito, nei confronti degli altri membri del gruppo. Il sistema quindi aggiornerà i loro **bilanci** generando i relativi **movimenti di denaro**.

Un **movimento di denaro** è caratterizzato da: un utente sorgente, un utente destinazione, l'importo di denaro e la data.



Una spesa può essere **commentata**  dai membri del gruppo coinvolto. Di ogni **commento** occorre conoscere: l'utente autore del commento, il testo e la data.

Il sistema mette a disposizione dei membri di uno stesso gruppo anche la possibilità di memorizzare una **lista di prodotti**  da comprare. La lista potrà essere consultata e aggiornata da ogni membro del gruppo, e servirà da promemoria per acquisti futuri.

Di ogni **prodotto** della lista è necessario specificare il nome e la quantità desiderata.



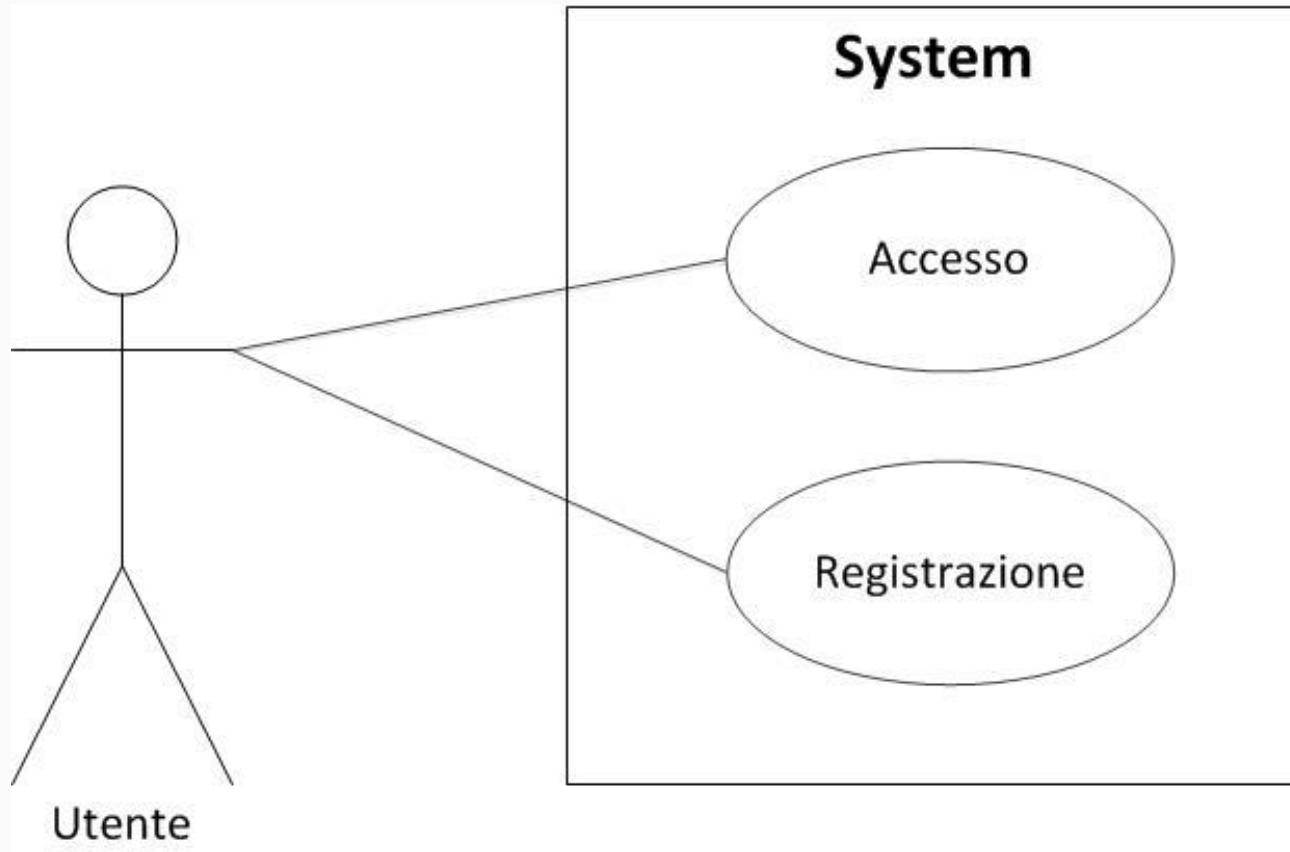
Ciascun utente può andare ad estinguere un eventuale debito con un amico effettuando un **saldo**. In questo caso dovrà specificare l'amico con il quale intende saldare il proprio debito, la cifra di denaro restituita e la data.

Il sistema deve inoltre essere in grado di mostrare all'utilizzatore un **riepilogo**. Il riepilogo potrà essere relativo a:

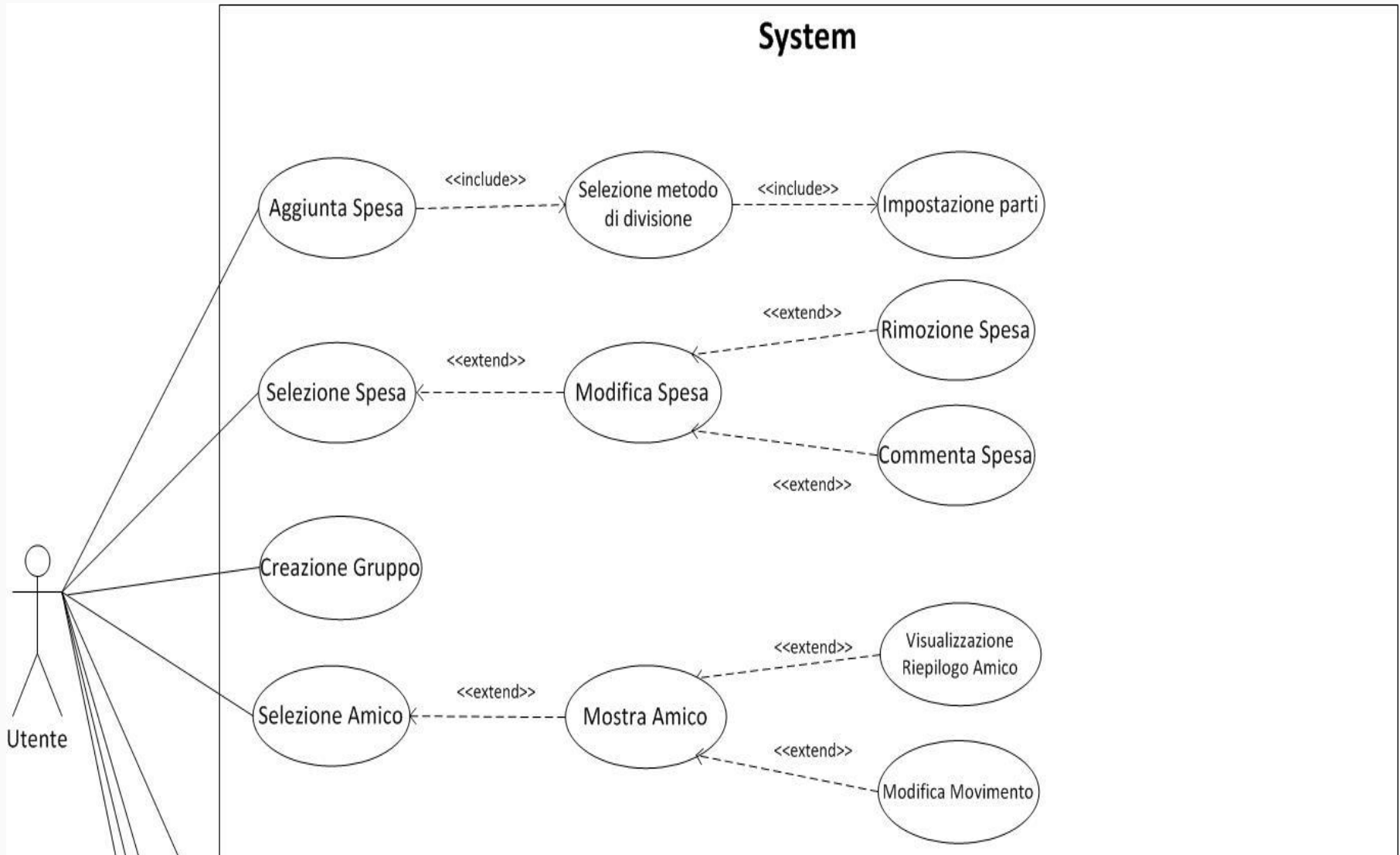
- tutti i gli amici dell'utente.
- tutti i gli altri membri di uno stesso gruppo di appartenenza.
- un singolo amico.

In ogni caso il riepilogo dovrà riportare al suo interno il **bilancio** e l'elenco delle **attività recenti**.

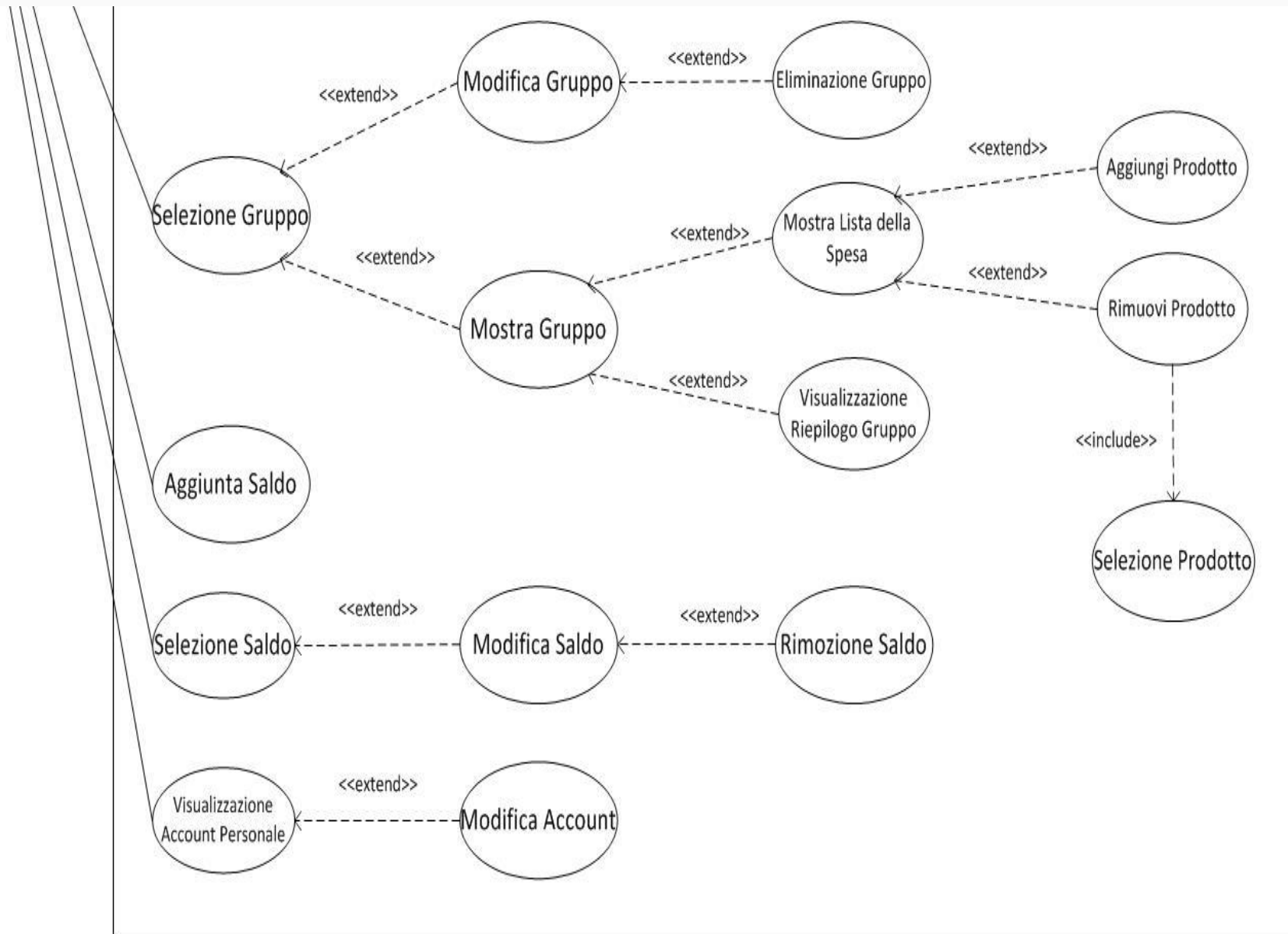
# Diagramma dei casi d'uso



# Diagramma dei casi d'uso



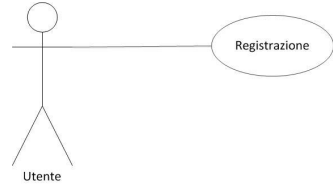




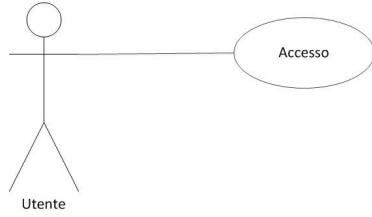
# Diagramma dei casi d'uso



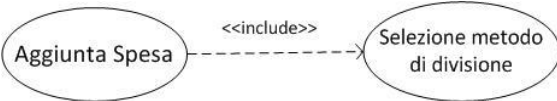
# Casi d'uso - Registrazione

Titolo		Registrazione
Descrizione	Registrazione di un utente nel sistema	
Relazioni	 <pre>graph LR; Utente((Utente)) --- Registrazione((Registrazione))</pre>	
Attori	Utente	
Precondizioni	Nessuna	
Postcondizioni	L'utente viene registrato nel sistema	
Scenario Principale	<ol style="list-style-type: none"><li>1. L'utente inserisce il proprio nome</li><li>2. L'utente inserisce il proprio cognome</li><li>3. L'utente inserisce l'email</li><li>4. L'utente inserisce la propria password due volte: la seconda per controllo</li><li>5. L'utente inserisce il telefono</li><li>6. L'utente inserisce la propria foto (default: <i>"Foto di default"</i>)</li><li>7. L'utente conferma l'inserimento e il sistema salva i dati</li></ol>	
Scenari Alternativi	Nessuno	
Requisiti non funzionali	La mail non deve essere già registrata nel sistema e la password deve contenere almeno 8 caratteri	
Punti aperti	Nessuno	

# Casi d'uso - Accesso

Titolo	Accesso
Descrizione	L'utente accede al sistema
Relazioni	 <pre>graph LR; Utente((Utente)) --- Accesso((Accesso))</pre>
Attori	Utente
Precondizioni	L'utente deve essere già registrato nel sistema
Postcondizioni	L'utente deve essere autenticato nel sistema
Scenario Principale	<ol style="list-style-type: none"><li>1. L'utente inserisce il proprio indirizzo email</li><li>2. L'utente inserisce la propria password</li><li>3. L'utente conferma i dati inseriti e il sistema verifica i dati</li></ol>
Scenari Alternativi	Nessuno
Requisiti non funzionali	Nessuno
Punti aperti	Nessuno

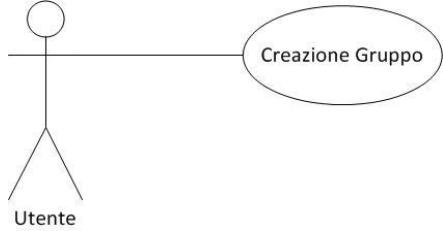
# Casi d'uso - Aggiungi Spesa

Titolo		Aggiungi Spesa
Descrizione	Creazione e registrazione di una nuova spesa nel sistema, in seguito a un acquisto effettuato dal gruppo.	
Relazioni		
Attori	Utente	
Precondizioni	<ul style="list-style-type: none"><li>• L'utente deve essere autenticato</li><li>• L'utente deve appartenere ad almeno un gruppo, con il quale poi condivide la spesa</li></ul>	
Postcondizioni	<ul style="list-style-type: none"><li>• La nuova spesa viene aggiunta al sistema</li><li>• I bilanci degli utenti coinvolti sono aggiornati</li></ul>	
Scenario Principale	<ol style="list-style-type: none"><li>1. L'utente seleziona il gruppo cui condivide la spesa</li><li>2. L'utente inserisce una descrizione</li><li>3. L'utente inserisce l'importo speso</li><li>4. L'utente sceglie l'utente pagante, tra i membri del gruppo selezionato</li><li>5. L'utente sceglie il metodo di divisione (default: <i>"Equo"</i>) e la data della spesa (default: <i>"Data odierna"</i>).</li><li>6. L'utente conferma i dati inseriti</li><li>7. Il sistema salva la spesa e genera i movimenti di denaro</li></ol>	
Scenari Alternativi	<p>5.a Se il metodo scelto richiede la definizione delle parti in cui dividere l'importo, verrà visualizzata una finestra per specificarle.</p>	
Requisiti non funzionali	L'importo deve essere maggiore di 0 e la data non può essere successiva alla data odierna	
Punti aperti	Si potrebbe aggiungere la possibilità di definire paganti multipli	

# Casi d'uso - Aggiungi Saldo

Titolo	Aggiungi Saldo
<b>Descrizione</b>	Aggiunta di un nuovo saldo nel sistema, in seguito ad un rimborso effettuato da un utente verso un altro.
<b>Relazioni</b>	<pre>graph LR     U[Utente] --- UC((Aggiunta Saldo))</pre> <p>The diagram shows a stick figure actor labeled 'Utente' connected by a line to an oval use case labeled 'Aggiunta Saldo'.</p>
<b>Attori</b>	Utente
<b>Precondizioni</b>	<ul style="list-style-type: none"><li>• L'utente deve essere autenticato</li><li>• Deve esistere un debito nei confronti di un altro utente amico</li></ul>
<b>Postcondizioni</b>	<ul style="list-style-type: none"><li>• Il sistema contiene un nuovo saldo</li><li>• I bilanci degli utenti coinvolti sono aggiornati</li></ul>
<b>Scenario Principale</b>	<ol style="list-style-type: none"><li>1. L'utente imposta l'utente sorgente del saldo</li><li>2. L'utente imposta l'utente destinatario del saldo</li><li>3. L'utente definisce l'importo</li><li>4. L'utente imposta la data</li><li>5. L'utente conferma i dati inseriti</li><li>6. Il sistema salva il nuovo saldo e genera il movimento di denaro</li></ol>
<b>Scenari Alternativi</b>	Nessuno
<b>Requisiti non funzionali</b>	<ul style="list-style-type: none"><li>• Sorgente e destinatario devono essere amici e utenti diversi (non si può saldare con se stessi.)</li><li>• L'importo deve essere maggiore di zero.</li></ul>
<b>Punti aperti</b>	Nessuno

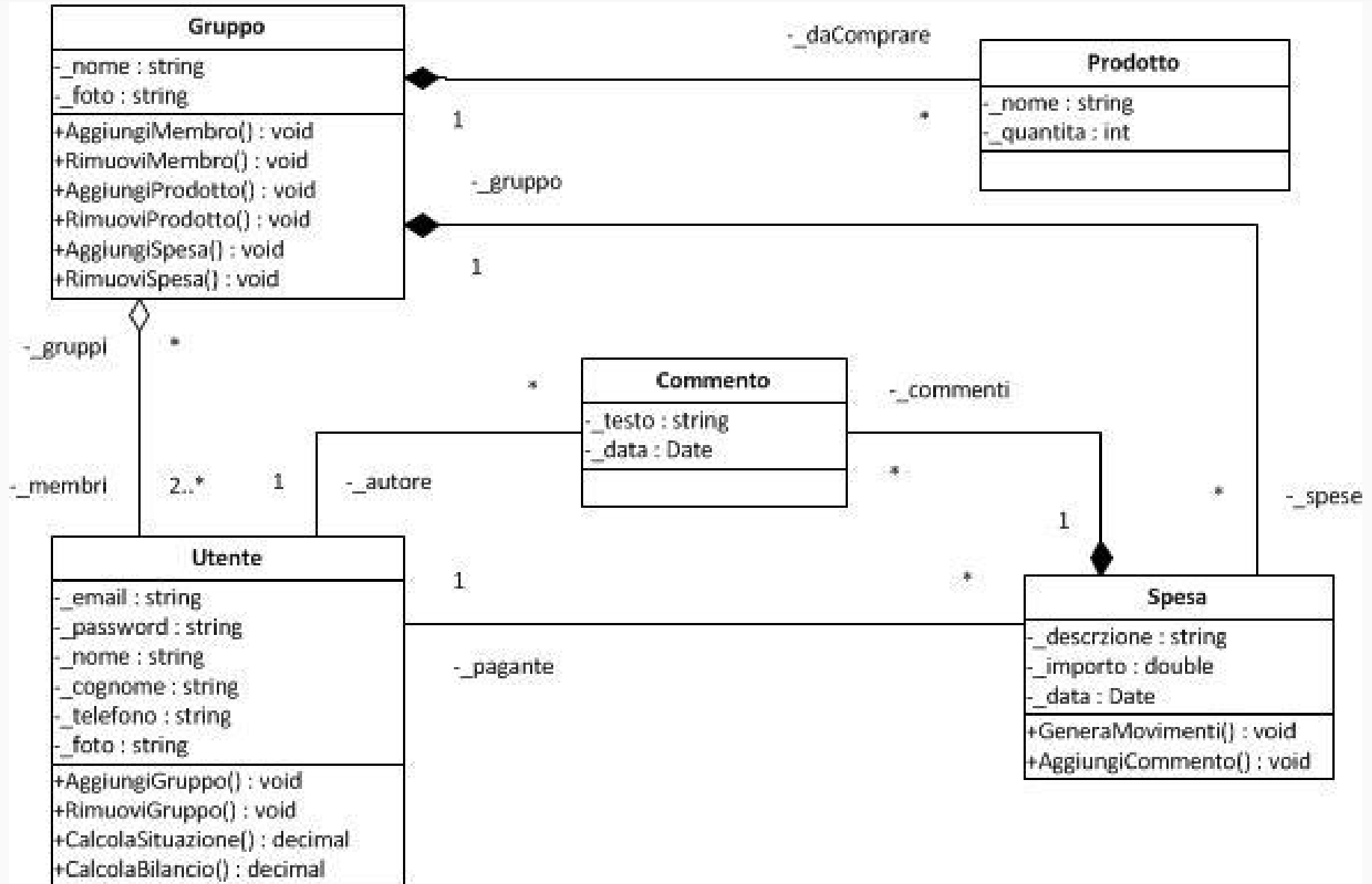
# Casi d'uso - Creazione Gruppo

Titolo	Creazione Gruppo
Descrizione	Aggiunta di un nuovo gruppo nel sistema
Relazioni	 <pre>graph LR; Utente((Utente)) --- CreazioneGruppo(Creazione Gruppo);</pre>
Attori	Utente
Precondizioni	L'utente deve essere autenticato
Postcondizioni	<ul style="list-style-type: none"><li>• Presenza di un nuovo gruppo nel sistema</li><li>• L'utente viene inserito nel gruppo appena creato</li></ul>
Scenario Principale	<ol style="list-style-type: none"><li>1. L'utente inserisce il nome del gruppo</li><li>2. L'utente aggiunge i membri inserendo per ciascuno l'indirizzo email</li><li>3. L'utente inserisce una foto del gruppo (default: <i>"Foto di default"</i>)</li><li>4. L'utente conferma i dati inseriti e viene aggiunto un nuovo gruppo nel sistema</li></ol>
Scenari Alternativi	Nessuno
Requisiti non funzionali	Deve essere aggiunto almeno un membro
Punti aperti	Nessuno

# Casi d'uso - Commenta Spesa

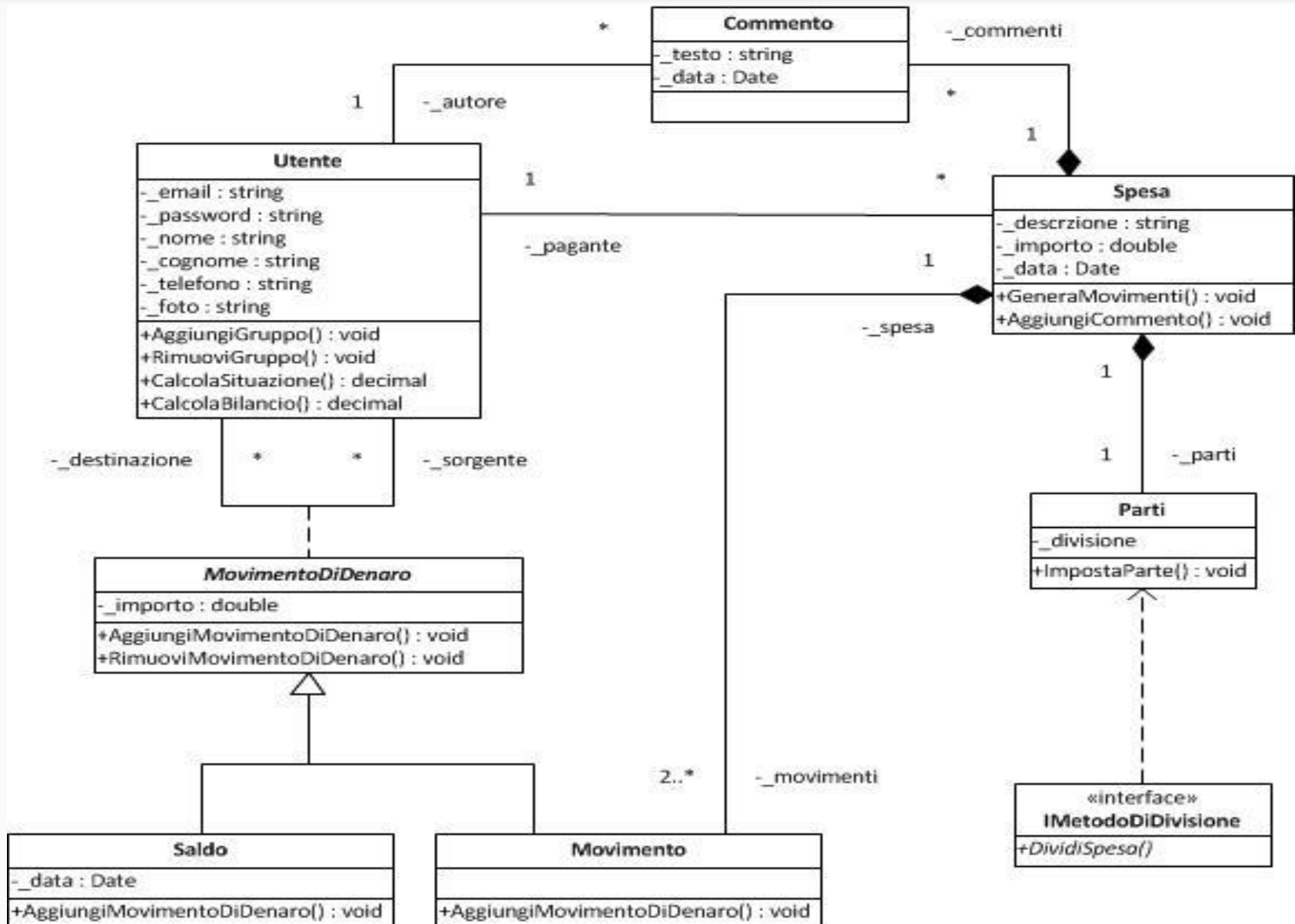
Titolo	Commenta Spesa
Descrizione	Aggiunta di commenti in una spesa
Relazioni	 <pre>graph LR; MS([Modifica Spesa]) -.-&gt; &lt;&lt;extend&gt;&gt;  CS([Commenta Spesa])</pre>
Attori	Utente
Precondizioni	<ul style="list-style-type: none"><li>• L'utente deve essere autenticato</li><li>• L'utente deve appartenere ad almeno un gruppo, con il quale è condivisa una spesa</li><li>• Deve esistere una spesa che coinvolga un gruppo del quale l'utente fa parte</li></ul>
Postcondizioni	<ul style="list-style-type: none"><li>• Presenza di un nuovo commento associato ad una spesa nel sistema</li></ul>
Scenario Principale	<ol style="list-style-type: none"><li>1. L'utente inserisce il commento</li><li>2. L'utente conferma il commento</li><li>3. Il commento viene inserito nel sistema in coda ai commenti già riguardanti la specifica spesa</li></ol>
Scenari Alternativi	Nessuno
Requisiti non funzionali	Il commento non può essere vuoto
Punti aperti	Nessuno

# Classi d'analisi

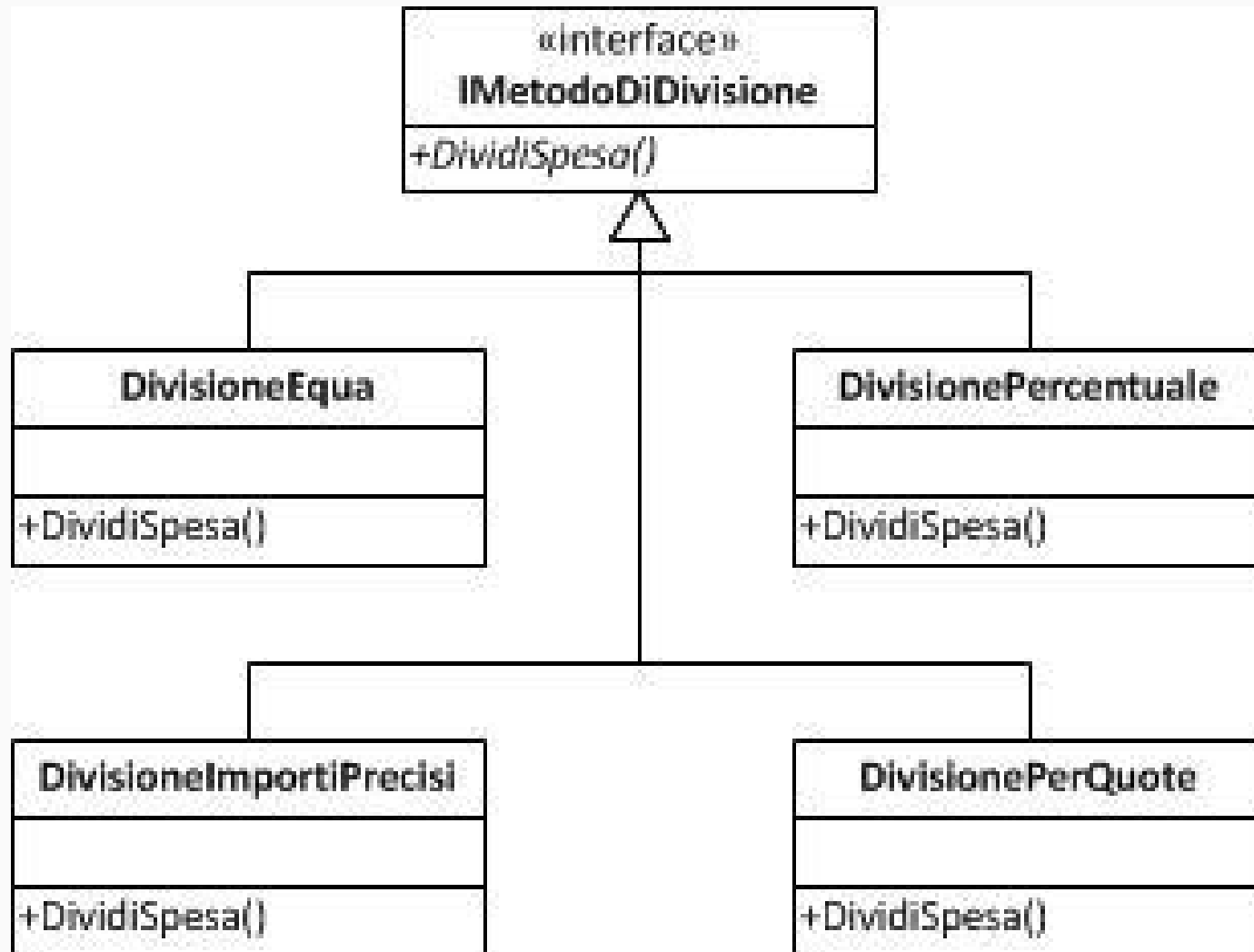




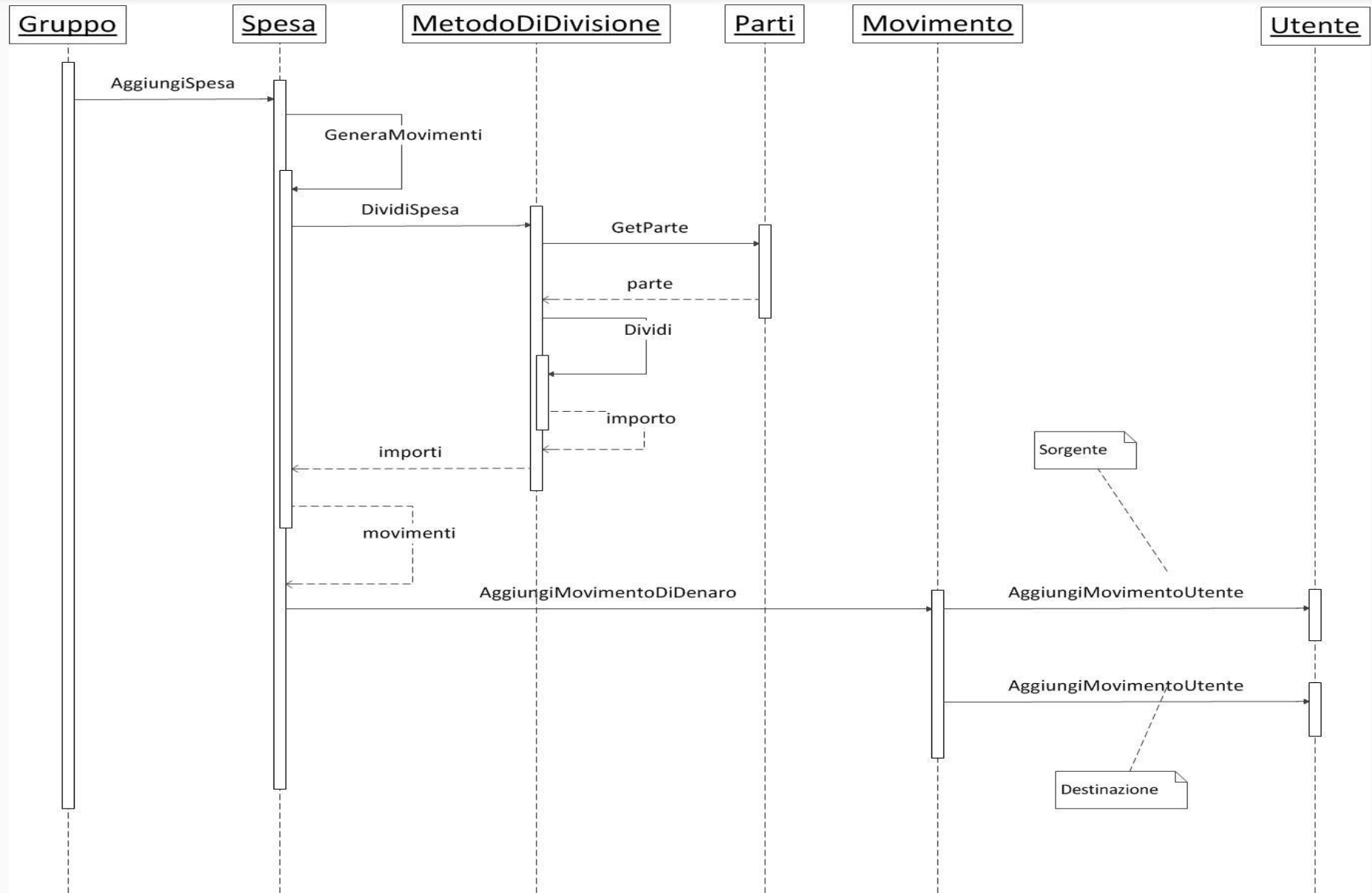
# Classi d'analisi



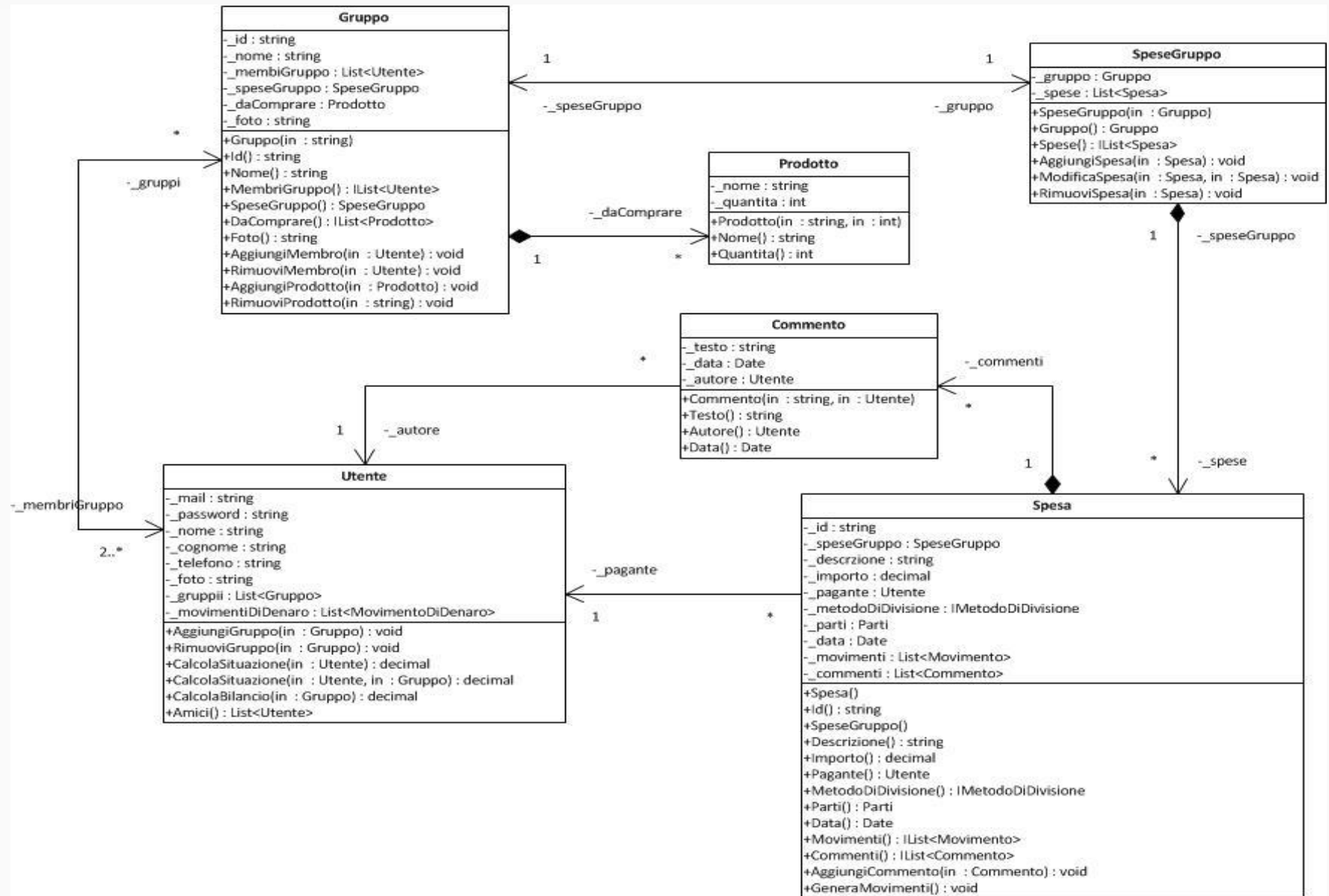
# Classi d'analisi



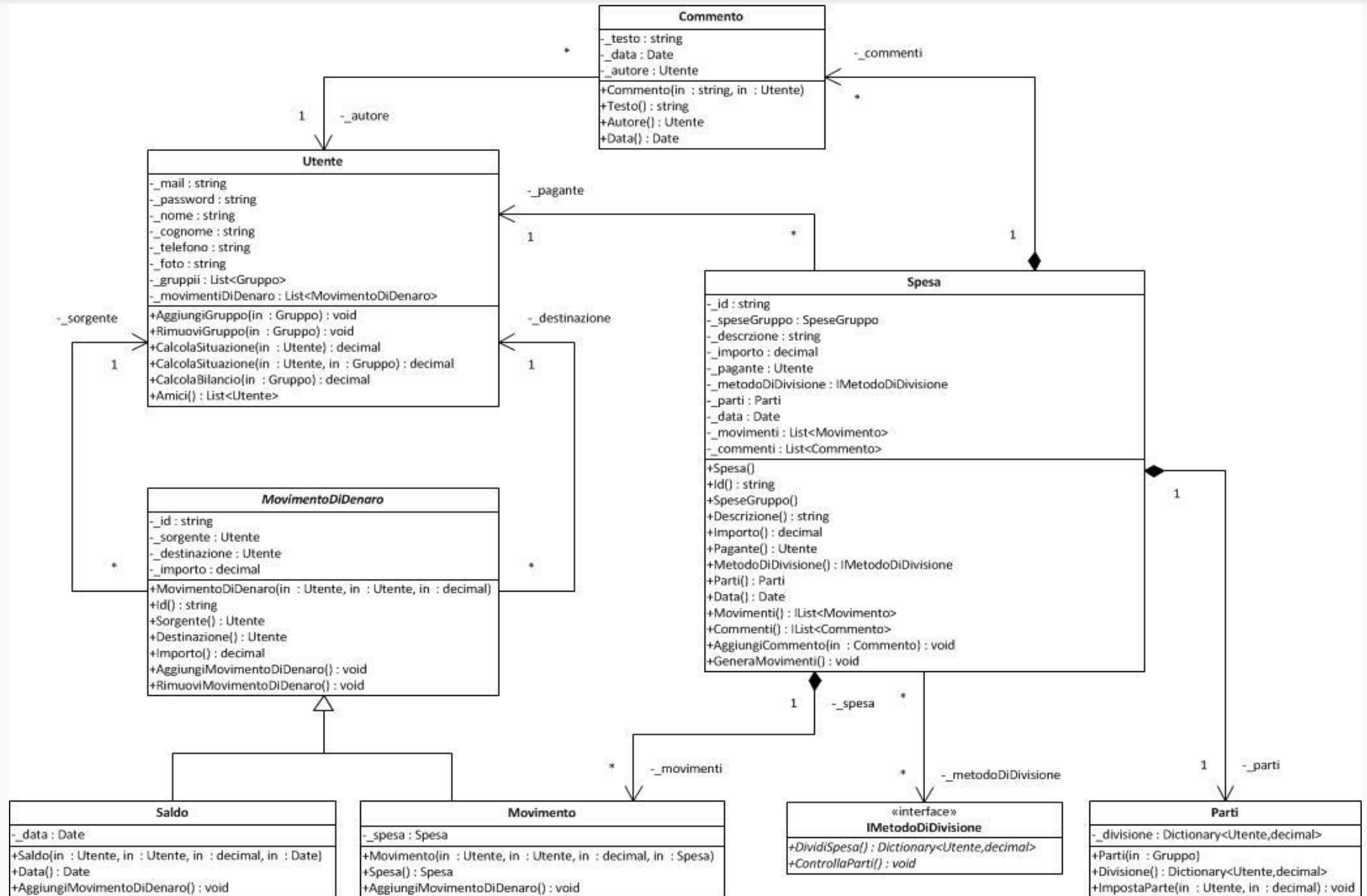
# Diagramma di sequenza



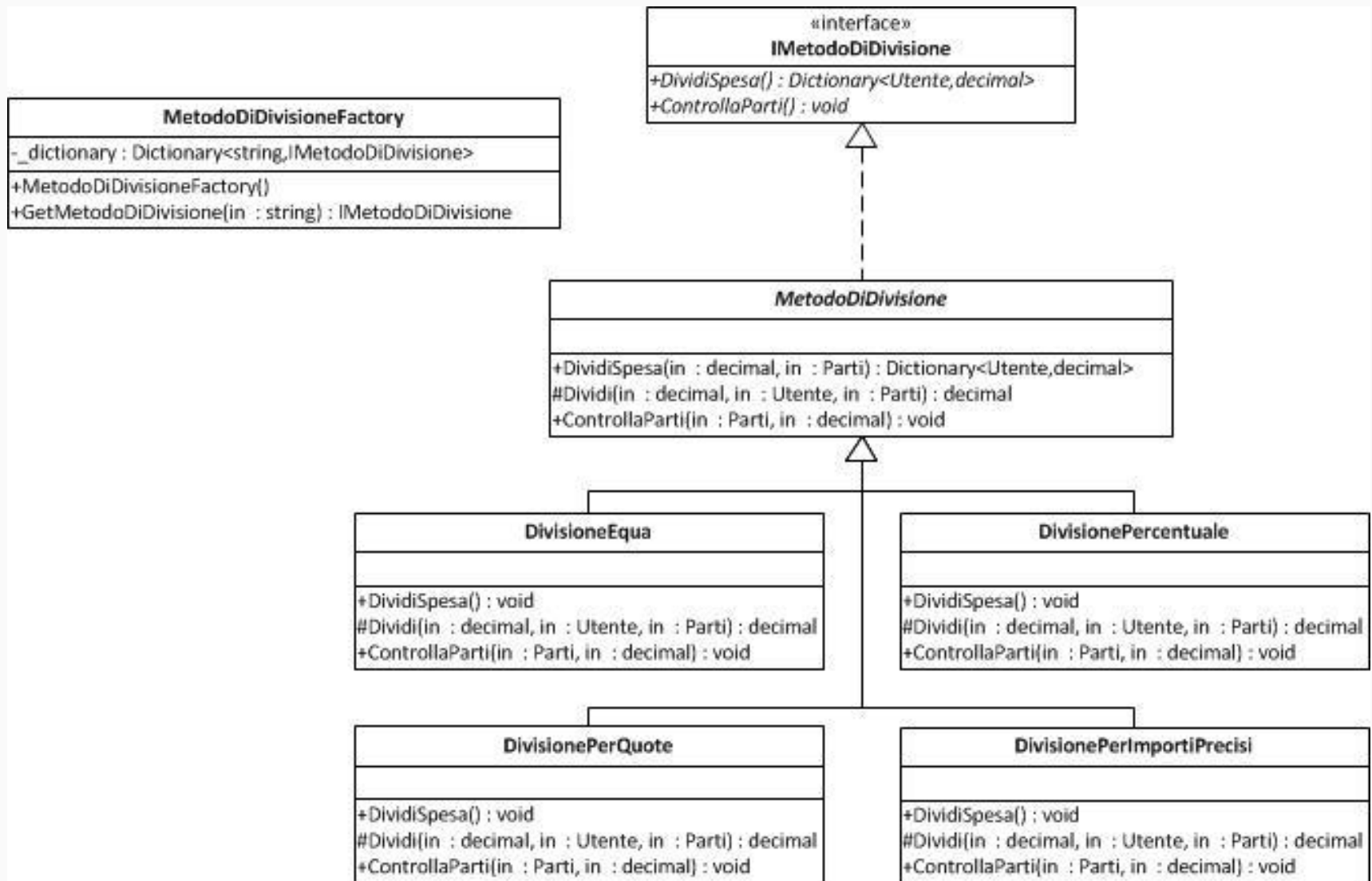
# Classi di Progettazione



# Classi di Progettazione



# Classi di Progettazione



# Design Principle

Progettando il nostro software abbiamo cercato di rispettare il più possibile i principi di progettazione.

Il Principio di **Single Responsibility** è stato applicato nella costruzione di ogni classe. In particolare creando la classe *Parti*, abbiamo voluto sgravare dal *MetodoDiDivisione* la responsabilità di gestire le assegnazioni e i controlli delle parti nelle quali la spesa viene divisa tra i vari utenti coinvolti.

Il caso dei *MovimentiDiDenaro* è invece un esempio di applicazione del **principio di sostituzione di Liskov**, in quanto tramite la classe astratta *MovimentoDiDenaro* il cliente è in grado di utilizzare correttamente le sottoclassi *Movimento* e *Saldo* senza accorgersene. Questo avviene grazie all'applicazione del **Design by Contract** implementato, per esempio, nel metodo *aggiungiMovimentoDiDenaro()*.

# Design Pattern

Abbiamo prestato particolare attenzione ai *metodi di divisione* delle spese, essendo una delle parti più delicate nella logica del programma. Applicando il pattern **Strategy** i *metodi di divisione* si sono mostrati più facili da intercambiare e il loro uso si è rivelato più semplice e dinamico, facilitando la possibile aggiunta di nuovi metodi di divisione.

Il nostro programma richiede anche di mostrare costantemente la situazione contabile, espressa dal riepilogo. Per far sì che ciò avvenga, nella view, abbiamo applicato il pattern **Observer**, rendendolo sensibile all'aggiunta di Spese e Saldi.

Infine abbiamo abbozzato un esempio di persistenza ( considerato che una buona progettazione della persistenza, richiederebbe un progetto a sè ) decidendo di renderla indipendente ed espandibile . L'interfaccia che si occupa della persistenza fa uso di tipi generici e definisce i metodi per caricare e salvare i dati generati dal programma. Sarà poi una classe che accederà ai database ad occuparsi di concretizzare le operazioni.