

Room8

Il team:

Sebastianelli Nicola - 0000722894

Sammaritani Gloria - 0000723439

Serafini Filippo - 0000723678

Documento dei requisiti

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**.

Glossario

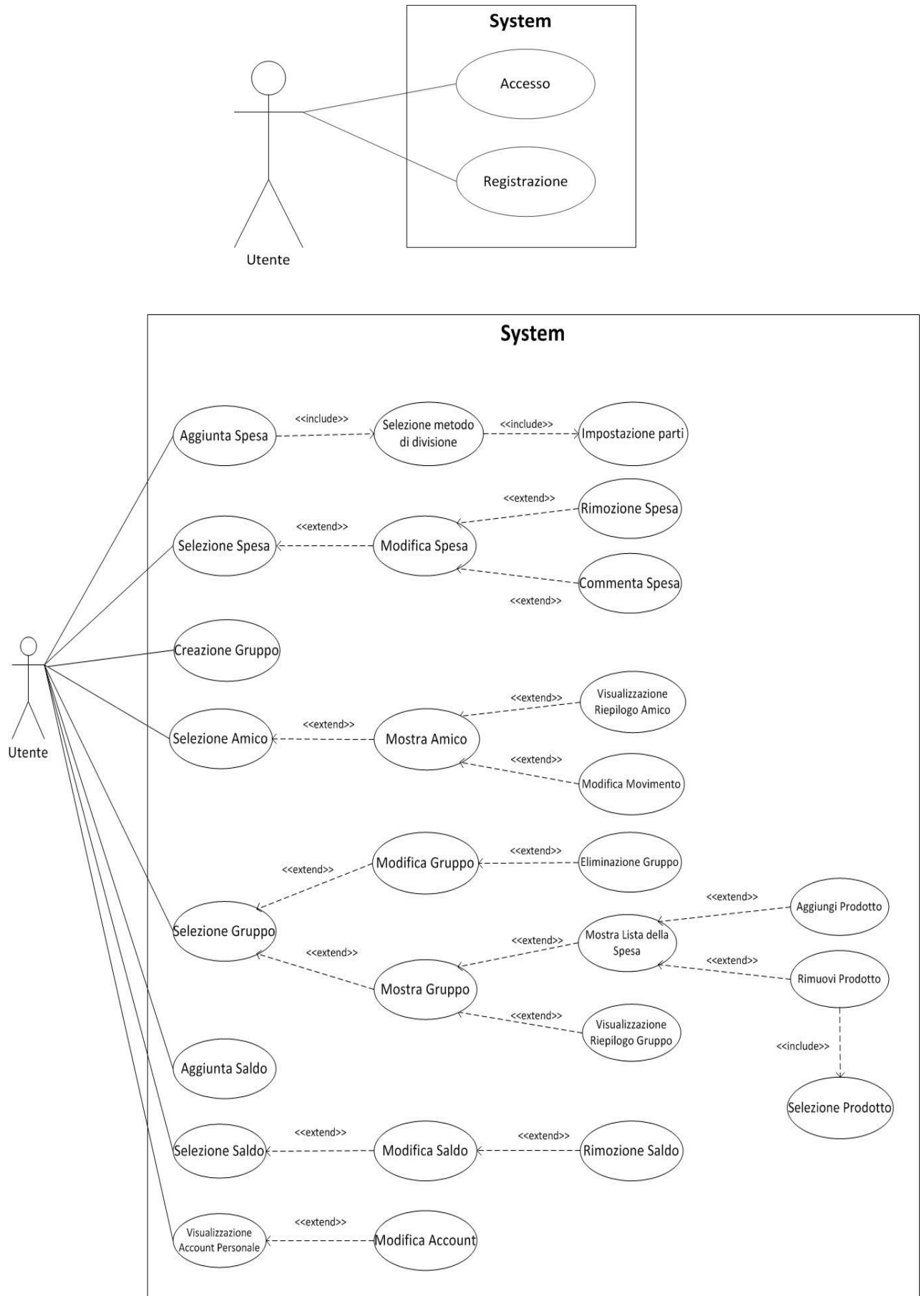
TERMINE	SIGNIFICATO E CARATTERISTICHE
Amico	Utente con cui si condivide l'appartenenza ad uno o più <i>gruppi</i> .
Attività recenti	Lista di <i>movimenti di denaro</i> che coinvolgono l' <i>utente</i> loggato, avvenuti in un determinato arco temporale.
Bilancio	Situazione contabile attuale con un <i>amico</i> .
Bilancio totale	Situazione contabile totale, derivata dalla somma di tutti i <i>crediti e debiti</i>
Commento	Commento lasciato da un <i>utente</i> relativamente a una <i>spesa</i> . Caratterizzato da: <ul style="list-style-type: none"> • Autore • Testo • Data
Credito	Somma di denaro che un <i>utente</i> deve ricevere da un altro.
Debito	Condizione inversa al <i>credito</i> . Somma di denaro che un <i>utente</i> deve dare ad un altro.
Gruppo	Insieme di più <i>utenti</i> , che condividono <i>spese</i> . Caratterizzato da: <ul style="list-style-type: none"> • Nome • Membri • Data di creazione • Foto (opzionale)
Metodo di divisione	Metodo attraverso il quale viene suddiviso l'importo della <i>spesa</i> tra i vari <i>utenti</i> coinvolti. Può essere: <ul style="list-style-type: none"> • In parti uguali • Per quote • Percentuale • Per importi precisi
Movimento	<i>Movimento di denaro</i> generato da una <i>spesa</i> .
Movimento di denaro	Flusso di denaro tra due <i>amici</i> . Caratterizzato da: <ul style="list-style-type: none"> • Sorgente • Destinazione • Importo • Data

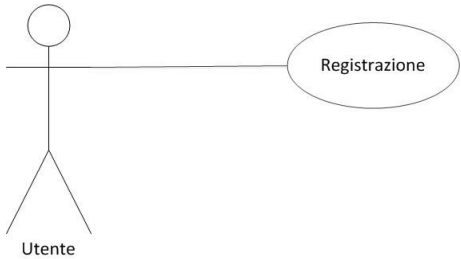
TERMINE	SIGNIFICATO E CARATTERISTICHE
Prodotto	Oggetto che si intende acquistare in una <i>spesa</i> futura. Caratterizzato da: <ul style="list-style-type: none"> • Nome • Quantità
Riepilogo	Visualizzazione di <i>bilancio</i> e <i>attività recenti</i> . Può essere: <ul style="list-style-type: none"> • Tra un utente e tutti gli utenti amici • Tra un utente e gli altri membri di uno stesso gruppo • Tra un utente e un singolo amico
Saldo	<i>Movimento di denaro</i> volto all'estinzione di un debito.
Spesa	Flusso di denaro che coinvolge i membri di un <i>gruppo</i> . Caratterizzato da: <ul style="list-style-type: none"> • Gruppo coinvolto • Descrizione • Importo speso • Utente pagante • Metodo di divisione • Data
Utente	Utilizzatore del software. Caratterizzato da: <ul style="list-style-type: none"> • Indirizzo email • Password • Nome e Cognome • Telefono • Foto (opzionale)

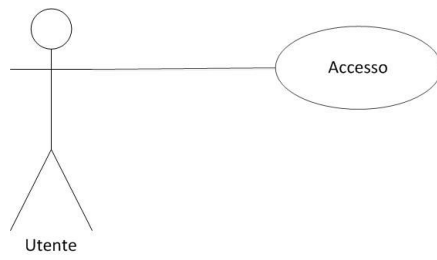
Casi d'uso

Note comuni:

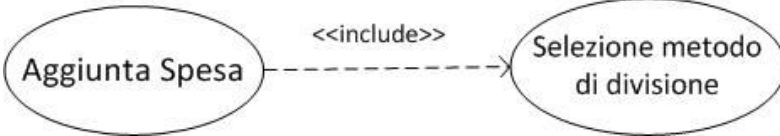
- L'utente è in grado di **annullare l'operazione corrente** in qualsiasi momento.
- Nel caso di **inserimento di un dato**:
 1. tra parentesi è indicato il **valore di default** (valore iniziale)
 2. in caso d'inserimento di un valore non ammesso, il sistema notifica l'errore e chiede di inserire un nuovo valore.



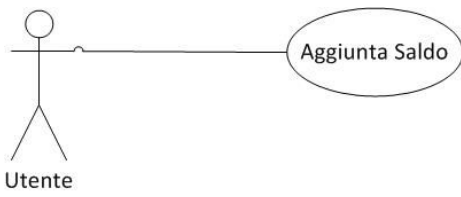
Titolo	Registrazione
Descrizione	Registrazione di un utente nel sistema
Relazioni	 <pre> graph LR Utente((Utente)) --- Registrazione([Registrazione]) </pre>
Attori	Utente
Precondizioni	Nessuna
Postcondizioni	L'utente viene registrato nel sistema
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente inserisce il proprio nome 2. L'utente inserisce il proprio cognome 3. L'utente inserisce l'email 4. L'utente inserisce la propria password due volte: la seconda per controllo 5. L'utente inserisce il telefono 6. L'utente inserisce la propria foto (default: "Foto di default") 7. L'utente conferma l'inserimento e il sistema salva i dati
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

Titolo	Accesso
Descrizione	L'utente accede al sistema
Relazioni	 <pre> graph LR Utente((Utente)) --- Accesso([Accesso]) </pre>
Attori	Utente

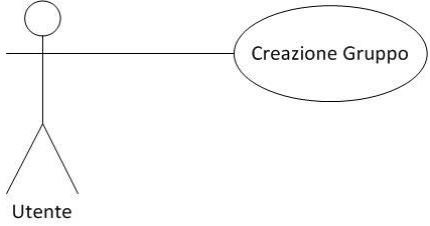
Precondizioni	L'utente deve essere già registrato nel sistema
Postcondizioni	L'utente deve essere autenticato nel sistema
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente inserisce il proprio indirizzo email 2. L'utente inserisce la propria password 3. L'utente conferma i dati inseriti e il sistema verifica i dati
Scenari Alternativi	Nessuno
Requisiti non funzionali	Nessuno
Punti aperti	Nessuno

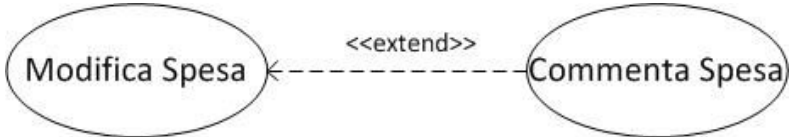
Titolo	Aggiungi Spesa
Descrizione	Creazione e registrazione di una nuova spesa nel sistema, in seguito a un acquisto effettuato dal gruppo.
Relazioni	 <pre> graph LR A([Aggiunta Spesa]) -.-> <<include>> B([Selezione metodo di divisione]) </pre>
Attori	Utente
Precondizioni	<ul style="list-style-type: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

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

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

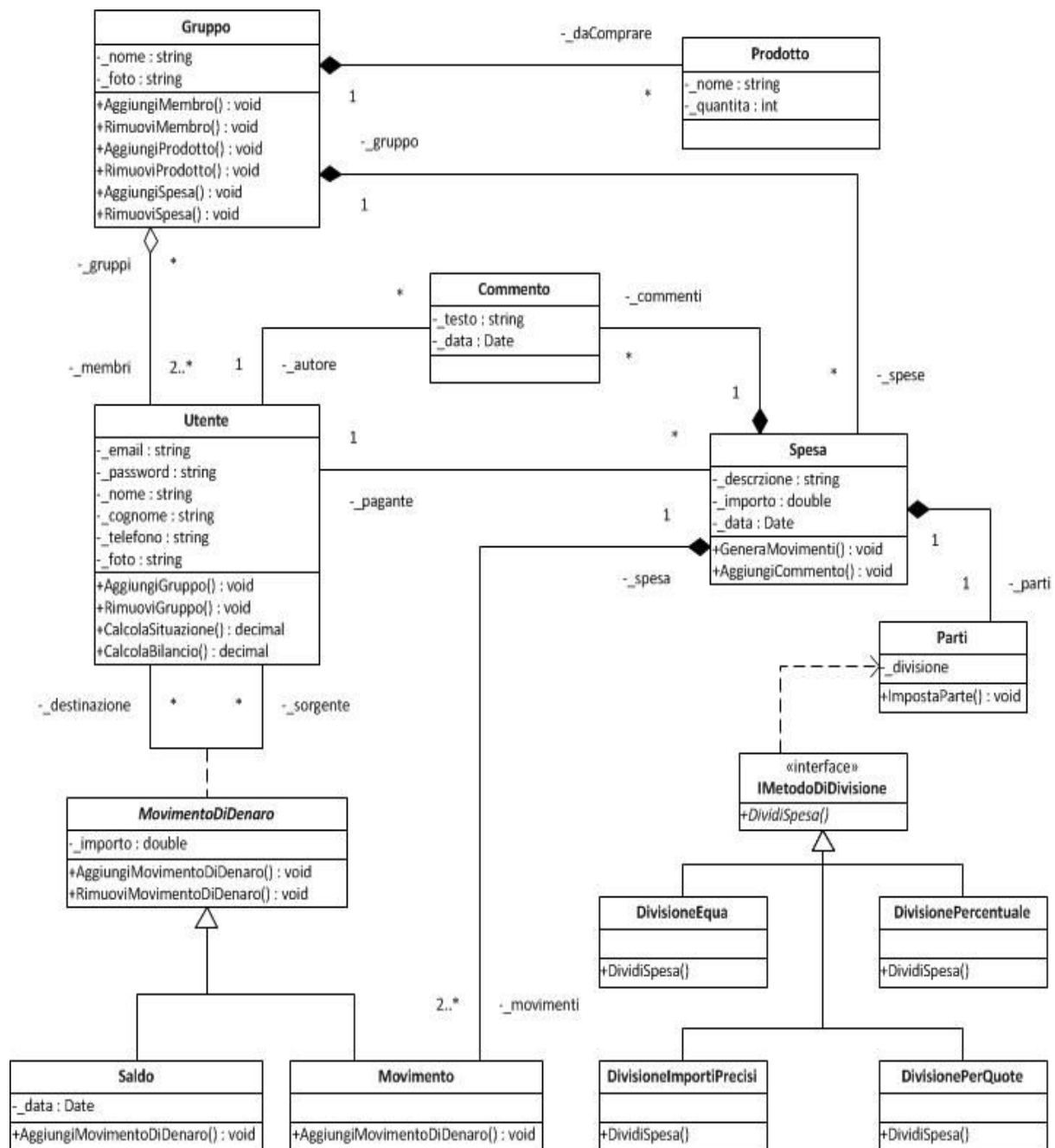
Titolo	Creazione Gruppo
Descrizione	Aggiunta di un nuovo gruppo nel sistema

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

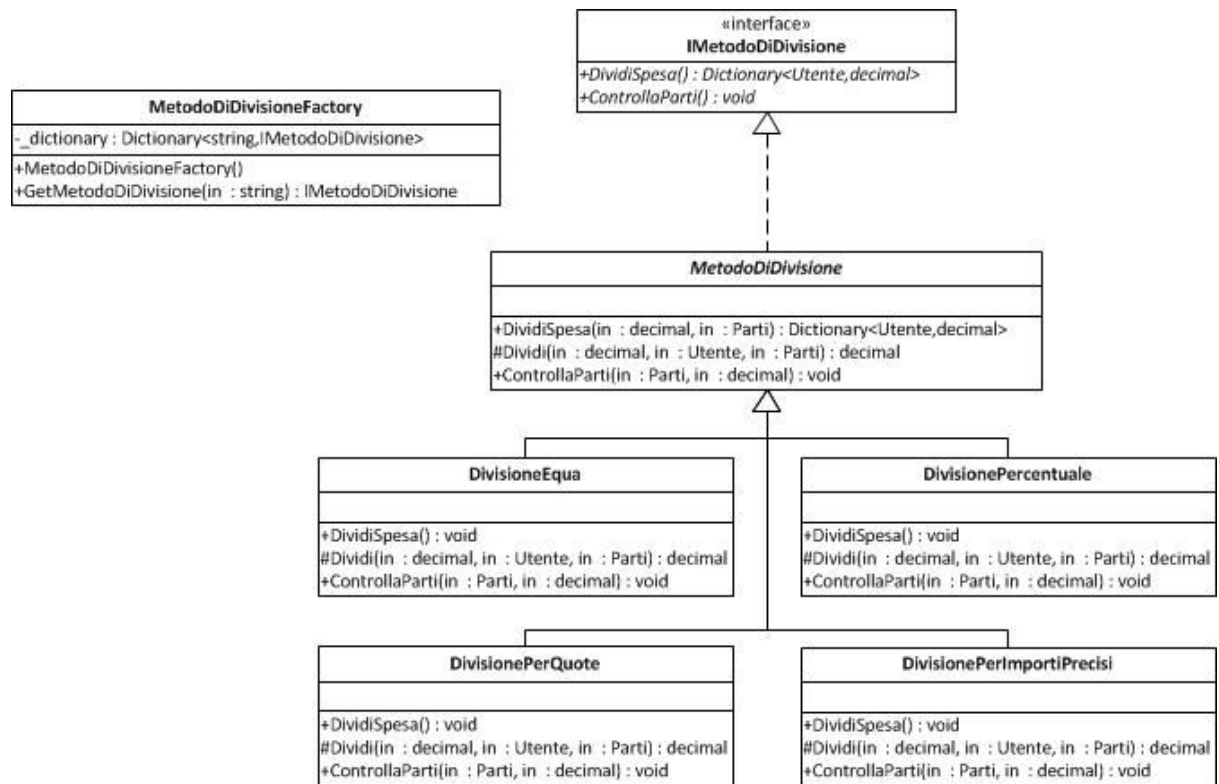
Titolo	Commenta Spesa
Descrizione	Aggiunta di commenti in una spesa
Relazioni	 <pre> graph LR CS((Commenta Spesa)) -.-> <<extend>> MS((Modifica Spesa)) </pre>
Attori	Utente
Precondizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato • L'utente deve appartenere ad almeno un gruppo, con il quale è condivisa una spesa • Deve esistere una spesa che coinvolga un gruppo del quale l'utente fa parte
Postcondizioni	<ul style="list-style-type: none"> • Presenza di un nuovo commento associato ad una spesa nel sistema
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente inserisce il commento 2. L'utente conferma il commento

	3. Il commento viene inserito nel sistema in coda ai commenti già riguardanti la specifica spesa
Scenari Alternativi	Nessuno
Requisiti non funzionali	Il commento non può essere vuoto
Punti aperti	Nessuno

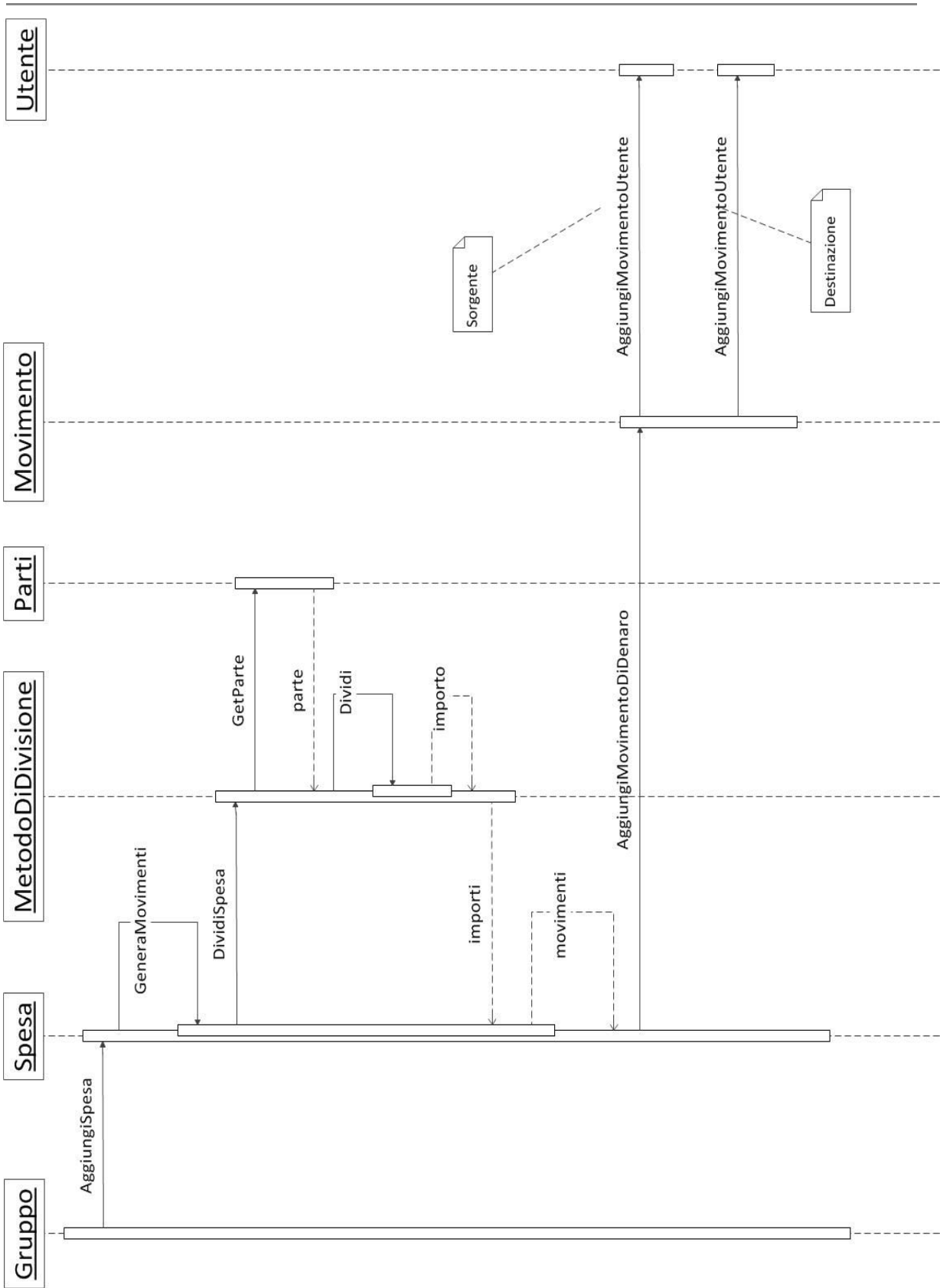
Classi di analisi







Modello dinamico



Design Pattern e Design Principles

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. Ad esempio la classe *Parti* sgrava dal *MetodoDiDivisione* la responsabilità di gestire le assegnazioni e i controlli delle parti nelle quali la spesa viene divisa, tra i vari utenti coinvolti.

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 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()*.

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.