Elaborazione - Iterazione 4

1 Introduzione

In questa quarta iterazione di Elaborazione si prendono in esame i seguenti requisiti:

1. La piattaforma ***invia delle notifiche*** ai follower degli utenti alla pubblicazione di nuovi post
2. E' possibile seguire gli account di altri whistleblower per visualizzare i post da loro pubblicati nella piattaforma, commentarli e ***mettere like***.

I casi d'uso di riferimento sono **UC9: Visualizza Notifiche**, **UC10: Inserisci Like**.

Nello specifico ci si concentra sulla:

- implementazione dello scenario principale di successo del caso d'uso **UC9** e sulle sue estensioni **1a**, **1b**, **3a**. In ordine, la **visualizzazione delle notifiche** ricevute dall'account di cui l'Utente è proprietario, la **cancellazione di una specifica notifica** (fornito il suo identificativo) e la **cancellazione di tutte le notifiche** presenti nell'account. Viene gestito anche il caso in cui non siano presenti notifiche da visualizzare (3a).
- implementazione dello scenario principale di successo per ciò che riguarda il caso d'uso **UC10**. Prendendo in considerazione anche l'estensione **1a**. Nell'ordine: "inserimento" e "rimozione" di un **like** da un post pubblico presente nella piattaforma.

Sono stati riscritti i casi d'uso **UC9** e **UC10** in formato dettagliato.

2 Analisi Orientata agli Oggetti

Nell'analisi orientata agli oggetti si fa uso degli elaborati: **Modello di Dominio**, **SSD** (Sequence System Diagram) e dei **Contratti delle operazioni**, al fine di descrivere il dominio da un punto di vista ad oggetti.

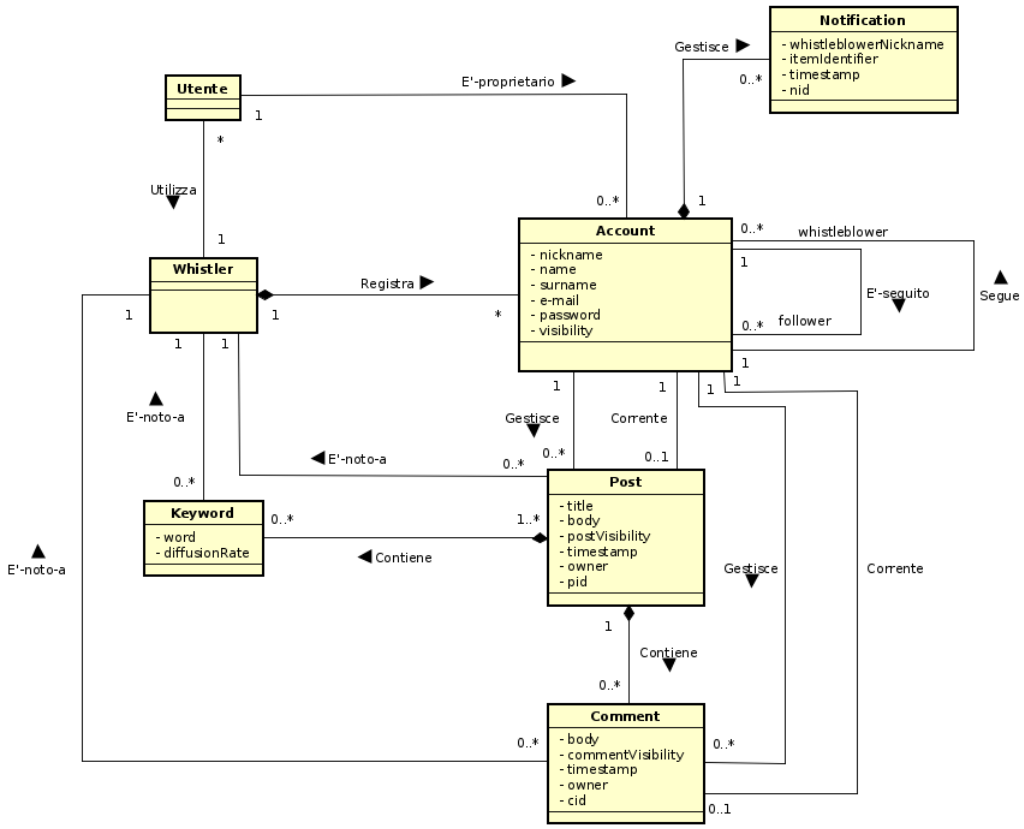
Modello di Dominio

La stesura del modello di dominio ci aiuta a decomporre il dominio in concetti o oggetti significativi. Tale elaborato rappresenta in modo visuale non solo le classi concettuali, ma anche le loro associazioni e gli attributi che le caratterizzano.

Prendendo in considerazione il caso d'uso **UC9** è possibile identificare la seguente classe concettuale:

- **Notifica:** comunica all'account dell'Utente la pubblicazione di un nuovo post da parte di un Whistleblower presente nella sua cerchia d'interesse.

Il modello di dominio ricavato è il seguente:



Prendendo in considerazione, invece, il caso d'uso **UC10** è possibile identificare la seguente classe concettuale:

- **Like:** è un simbolo che è possibile apporre a ciascun post pubblico in segno di sostegno al contenuto del post.

Il modello di dominio ricavato è il seguente:

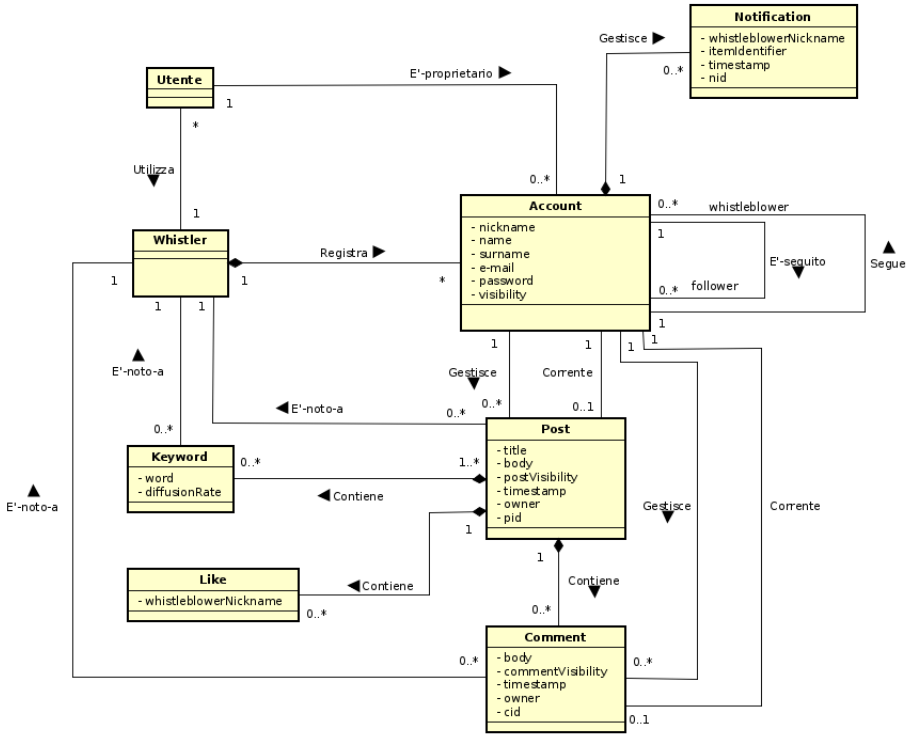
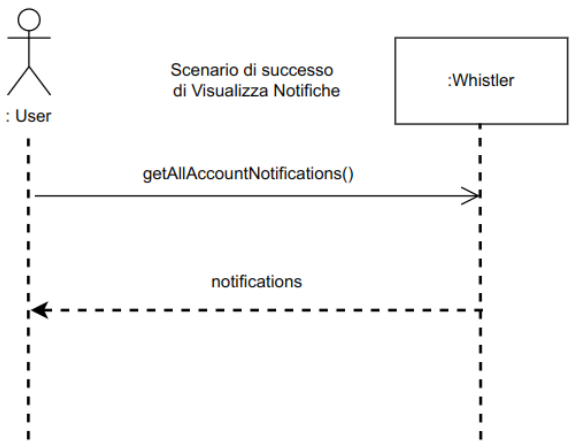


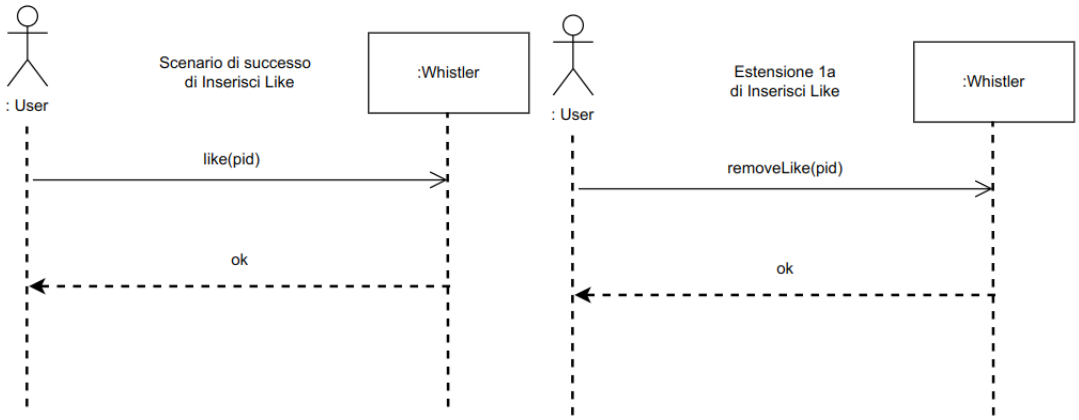
Diagramma di sequenza di sistema (SSD)

I diagrammi di sequenza di sistema, sono degli elaborati che mostrano gli eventi di input e di output relativi al sistema in discussione (a scatola nera).
Di seguito si riportano gli SSD ottenuti analizzando i casi d'uso presi in esame, sia nei loro **scenari principali di successo** che negli **scenari alternativi** più frequenti o complessi.

Per ciò che riguarda il caso d'uso **UC9**:



Per il caso d'uso **UC10**:



(a) scenario di successo **UC10** inserisci like (b) estensione **UC10_1a** rimuovi like

Contratti delle Operazioni

Di seguito si riportano i contratti delle operazioni di sistema ritenuti di maggiore rilevanza e identificati mediante l'analisi degli SSD precedentemente elaborati. I contratti delle operazioni permettono di fornire maggiori dettagli sull'effetto delle operazioni di sistema.

Contratto CO1: clearNotification
Operazione : clearNotification(nid:String)
Riferimenti : caso d'uso: Visualizza Notifiche
Pre-condizioni :
- l'Utente possiede un account ed è autenticato nella piattaforma;
- nell'account dell'Utente è presente una notifica n avente identificativo uguale al nid fornito;

Post-condizioni :
- è stata rimossa l'associazione "Gestisce" tra notifica ed account;
- n è stata rimossa.

Contratto CO2: clearAllNotifications
Operazione : clearAllNotifications()
Riferimenti : caso d'uso: Visualizza Notifiche
Pre-condizioni :
- l'Utente possiede un account ed è autenticato nella piattaforma
- nell'account dell'Utente è presente almeno una notifica.

Post-condizioni :
- tutte le associazioni "Gestisce" tra le notifiche presenti e l'account sono state rimosse;
- tutte le istanze di notifica associate all'account sono state rimosse.

Contratto CO3: like
Operazione : like(pid:String)
Riferimenti : caso d'uso: Inserisci Like
Pre-condizioni :
- l'Utente possiede un account ed è autenticato nella piattaforma
- nel Sistema è presente almeno un post pubblico visibile all'Utente
Post-condizioni :
- è stato aggiunto il nickname dell'account che appone il like nella lista dei likes del post (avente identificatico uguale a pid)
- è stata creata l'associazione "Contiene" tra post e like.

Contratto CO4: removeLike
Operazione : removeLike(pid:String)
Riferimenti : caso d'uso: Inserisci Like
Pre-condizioni :
- l'Utente possiede un account ed è autenticato nella piattaforma
- è presenta almeno un post in cui l'account dell'Utente ha precedentemente inserito un like.
Post-condizioni :
- il nickname dell'account è stato rimosso dalla lista dei likes del post.
- l'associazione "Contiene" tra post e like è stata rimossa;

3 Progettazione

Dopo aver terminato l'analisi orientata ad oggetti per questa iterazione, sfruttando gli elaborati prodotti, si passa alla fase di progettazione. Di seguito si riportano i Diagrammi di Interazione (nello specifico **Diagrammi di Sequenza**) - modellazione dinamica - ed il **Diagramma delle classi** - modellazione statica, tra loro complementari.
Durante la stesura di quest'ultimi sono stati tenuti a mente ed applicati i vari principi di progettazione OO, quali i **patter GRASP** per l'assegnazione delle responsabilità ed i **design pattern Gang-of-Four (GoF)**.

Diagrammi di Sequenza

Affinché il caso d'uso **UC9 - Visualizza Notifiche** sia possibile è necessario che venga inviata una notifica a tutti i Followers di un determinato Account ogni volta che quest'ultimo pubblica un Post avente Visibilità Pubblica.
Si è scelto di utilizzare il **Pattern GoF - Observer** per la gestione della notifica.

Nello specifico si è deciso di utilizzare l'implementazione tramite l'interfaccia **PropertyChancheListener** essendo ormai sconsigliata l'implementazione tramite la libreria *java.util*, poiché le interfacce ***java.util.Observer*** e ***java.util.Observable*** risultano deprecate. La motivazione di ciò, in breve, consiste nel fatto che tali inter-

facce non forniscono un modello ad eventi sufficientemente ricco per le applicazioni. E' possibile infatti venire a conoscoscenza del fatto che "qualcosa" sia cambiato in un oggetto, ma non effettivamente "il cosa" sia cambiato nello specifico. Alternativa valida è l'implementazione tramite i **Listeners** che risolvono tale problema.

Nel caso in esame la classe Account risulta essere sia un **Observable** che un **Observer**.
Per rendere la classe Account un Observable è necessario che essa:

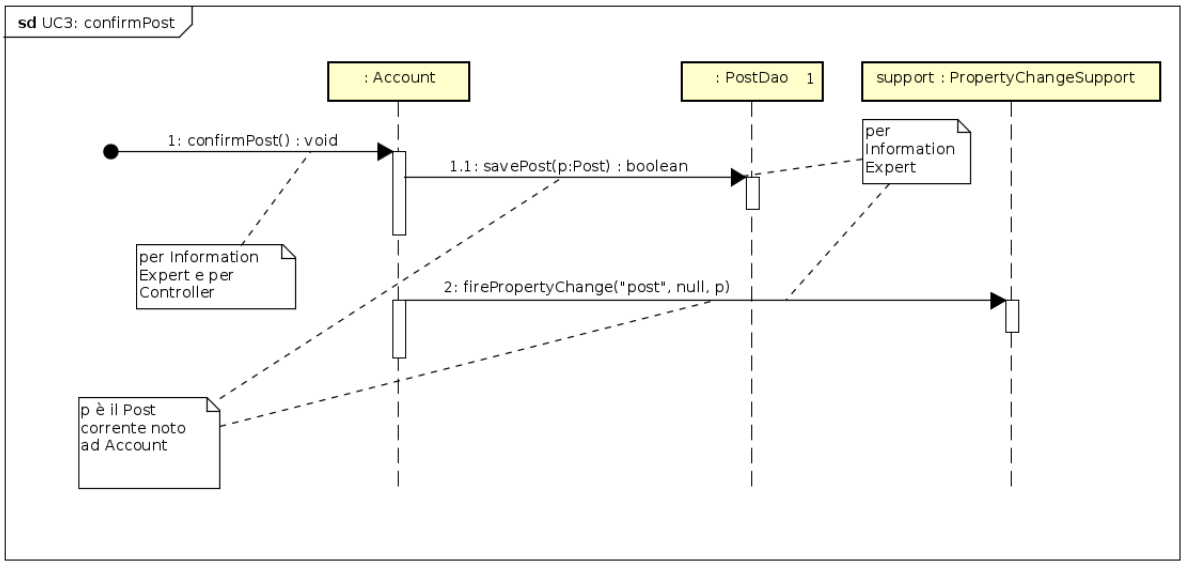
- presenti l'attributo ***support***
- implementi il metodo **addPropertyChangeListener**
  (necessario per l'aggiunta di Observers ad un Observable)
- implementi il metodo **removePropertyChangeListener**
  (necessario per la rimozione di Observers da un Observable)
- notifichi gli *Observers* richiamando il metodo **firePropertyChange** del ***support*** (all'interno del metodo *confirmPost()*, nel caso in esame).

Per rendere, invece, la classe Account un Observer è necessario che essa:

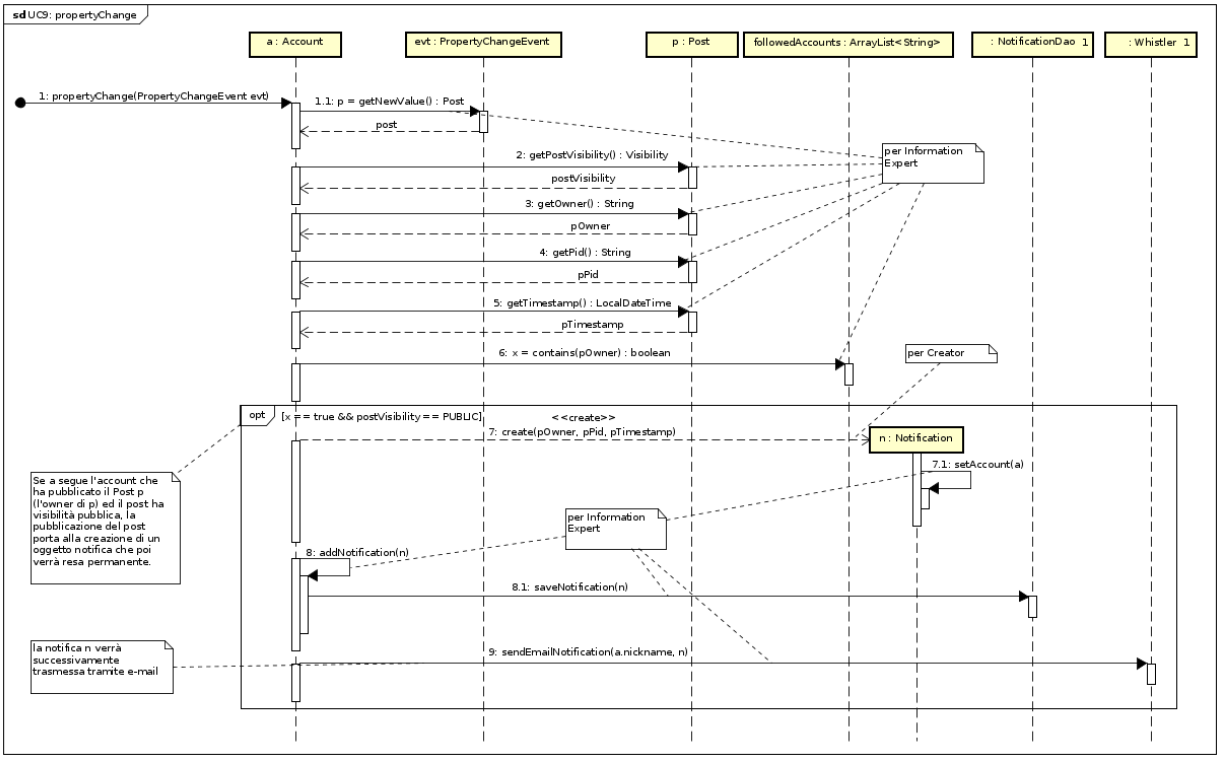
- implementi l'interfaccia ***PropertyChangeListener***
- implementi il metodo ***propertyChange***
  (che viene richiamato ogniqualvolta l'Observer riceve una notifica dall'Observable a cui è registrato)

Si riportano di seguito i diagrammi di sequenza del metodo **confirmPost()** aggiornato (caso d'uso **UC3 - Gestisci Post**) e l'implementazione del metodo **propertyChange**.

- Conferma di un Post

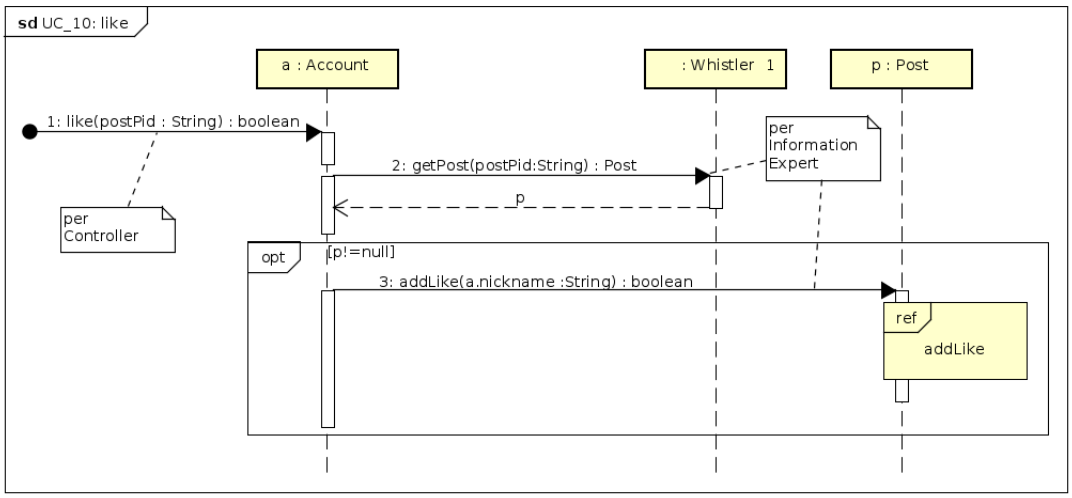


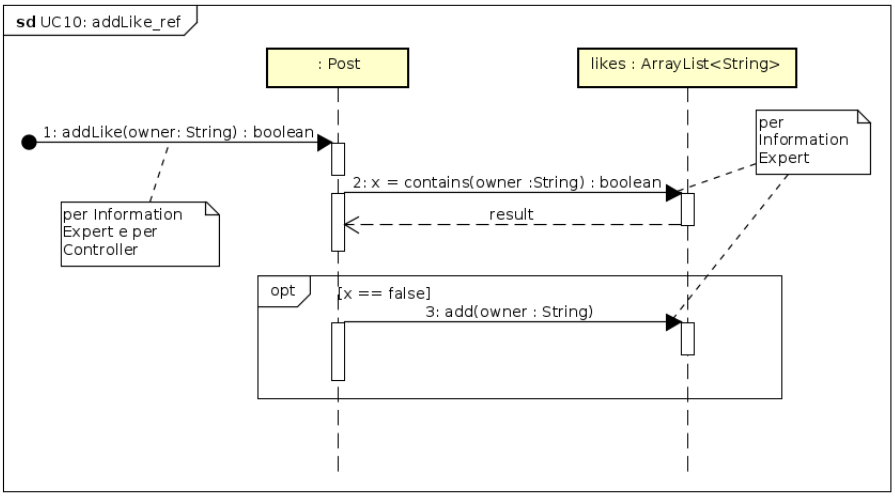
- Metodo chiamato lato follower alla pubblicazione di un nuovo Post



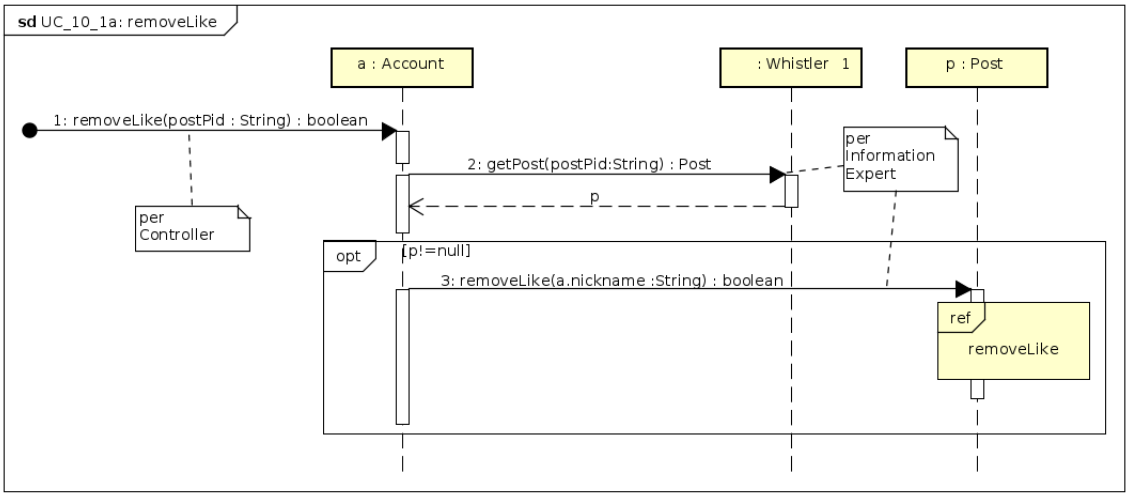
Si noti come la Notifica una volta resa permanente viene trasmessa anche per e-mail sfruttando l'indirizzo e-mail associato all'Account.

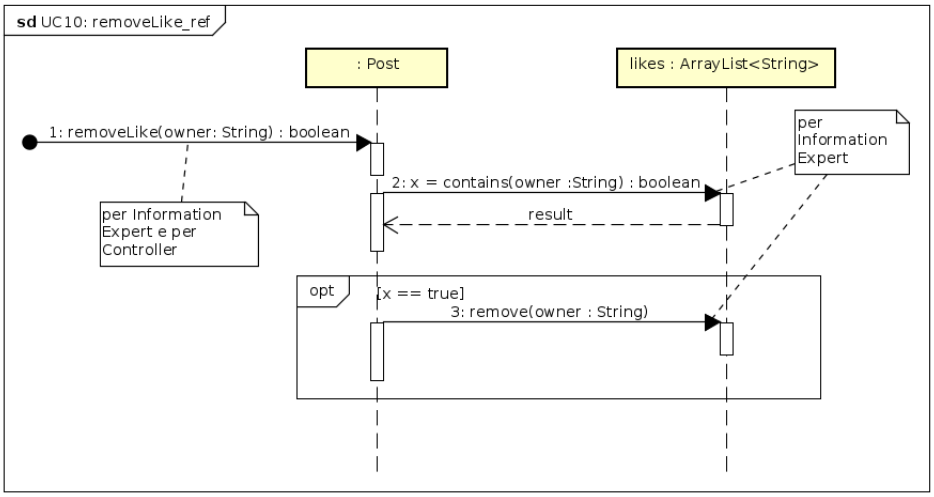
- Aggiunta di un Like





- Rimozione di un Like





Diagrammi delle Classi

- DCD : Elaborazione 4
Si rimanda al file **Whistler_E4.asta** nella cartella 05_Elaborazione_4 per la consultazione del diagramma delle Classi relativo all'elaborazione 4.

Testing

Si riportano di seguito i test unitari che sono stati aggiunti durante l'elaborazione corrente.

Test Unitario

La strategia adottata consiste nel definire una classe di test per ciascuna classe da verificare. Questa classe di test contiene uno o più metodi di test per ciascun metodo pubblico della classe da verificare.
Fanno eccezione i metodi introdotti dall'implementazione del caso d'uso **UC9** per i quali si è deciso di realizzare un unico metodo di test complessivo.

Classi di test:

Account Class
- 1) testNotification **(CF)**
- 2) testClearNotification **(CF)**
- 3) testClearAllNotifications **(CF)**
- 4) testLike_likeNotPresent **(CE) (CF)**
- 5) testLike_likeAlreadyPresent **(CE) (CF)**
- 6) testRemoveLike_likePresent **(CE) (CF)**
- 7) testRemoveLike_likeNotPresent **(CE) (CF)**

Post Class
- 1) testAddLike_likeNotPresent **(CE) (CF)**
- 2) testAddLike_likePresent **(CE) (CF)**
- 3) testRemoveLike_likePresent **(CE) (CF)**
- 4) testRemoveLike_likeNotPresent **(CE) (CF)**

Sono stati individuati i casi di test tenendo a mente la definizione di **Black-box Testing** (Funzionale), ovvero la determinazione dei casi di test sulla base della specifica di ciascun componente, non tenendo conto della sua struttura interna. I **dataset** usati sono stati scelti tenendo a mente le seguenti tecniche:
- **tecnica di copertura delle classi di equivalenza (CE)**
- **tecnica di analisi dei valori estremi (VE)**
- **tecnica di copertura delle funzionalità (CF)**

Per ciascun metodo di test elencato in precedenza si è indicato l'acronimo della tecnica di riferimento adottata accanto al suo nome.

Test di Sistema

Si è scelto di eseguire dei test di sistema manuali, al fine di testare la struttura dell'interfaccia utente a caratteri - basata su console - provando a portare a termine i requisiti funzionali indicati nell'iterazione corrente.

A seguito dei **test di Unità e di Sistema** sono state apportate opportune modifiche per rendere più robusta la gestione degli input inseriti dall'utente, durante l'interazione con il Sistema, e per migliorare il comportamento dei metodi presi in esame.

Test di Regressione

A valle di ogni modifica sono stati eseguiti i test unitari e di sistema, per verificare di non aver introdotto ulteriori e nuovi difetti.