Elaborazione - Iterazione 1

1 Introduzione

Rivisti i documenti della fase di ideazione, sono state apportate delle modifiche nella numerazione dei casi d'uso precedentemente individuati, tenendo conto della maggior importanza di ciascuno dal punto di vista dell'architettura e del rischio. E' stato scritto, inoltre, il caso d'uso UC1 in formato dettagliato.
Lo scopo di questa prima iterazione di Elaborazione è quello di implementare i seguenti requisiti:

1. Ogni utente possiede un account identificato da un nickname ed associato ad un profilo.

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 **UC1: Gestisci Account** e **UC2: Segui Account**. Nello specifico ci si concentra sulla:

- implementazione dello scenario principale di successo per ciò che riguarda il caso d'uso UC1, tralasciando in questa iterazione, per semplicità lo scenario alternativo di rimozione dell'account. Si rimanda, inoltre, l'implementazione del profilo ad iterazioni successive.

- implementazione dello scenario principale di successo per ciò che riguarda il caso d'uso UC2, tralasciando in questa iterazione, per semplicità, lo scenario alternativo di rimozione degli account seguiti dalla cerchia d'interesse. Si rimanda, inoltre, l'integrazione con la gestione dei post, della home e delle notifiche ad iterazioni successive nelle quali verranno presi in considerazione i rispettivi casi d'uso.

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.

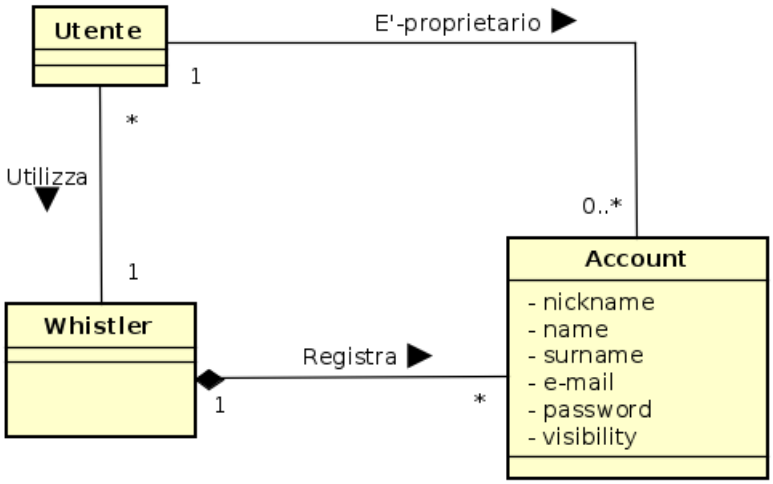
Modello di Dominio

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.

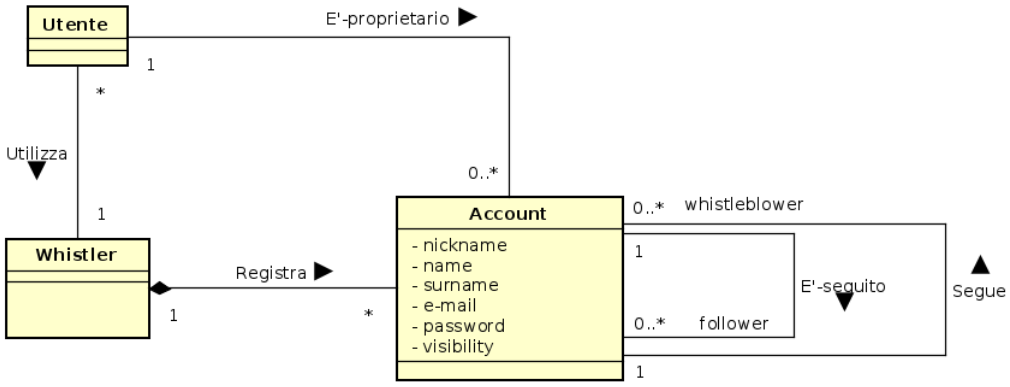
Prendendo in esame il caso d'uso **UC1** è possibile identificare le seguenti classi concettuali:

- **Utente:** è l'attore primario, rappresenta l'utilizzatore del Sistema
- **Whistler:** rappresenta il Sistema ovvero la piattaforma di microblogging Whistler
- **Account:** rappresenta l'identità dell'Utente all'interno della piattaforma di microblogging. Contiene tutte le informazioni personali, i post, le impostazioni ed i follower dell'Utente.

Il modello di dominio ricavato è il seguente:



Prendendo in esame, invece, il caso d'uso **UC2** emergono le due **associazioni riflessive** visibili nel modello di dominio aggiornato:

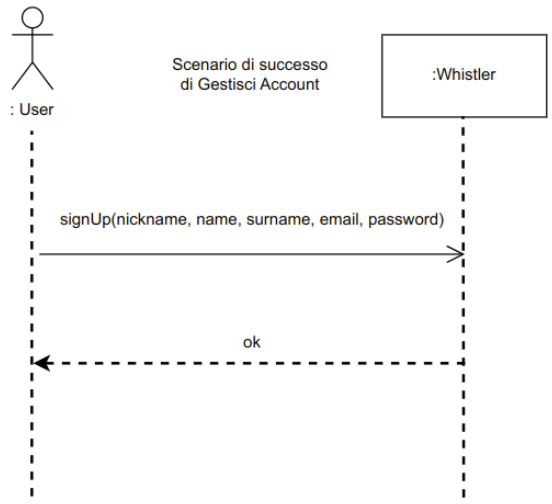


Tali associazioni, grazie alla presenza dei **ruoli** associati alle **molteplicità**, evidenziano che un Account può essere seguito da 0 o più Account in qualità di Followers e che questo a sua volta può seguire 0 o più Account (Whistleblowers).
E' importante, inoltre, sottolineare che un Account può essere seguito non solo da Accounts che esso stesso segue (quindi presenti nella sua cerchia d'interesse), ma anche da Accounts che non vengono seguiti a loro volta. Questo ha portato a separare, quella che potrebbe apparire in un primo momento, un'unica associazione riflessiva in due associazioni riflessive distinte e saparate.

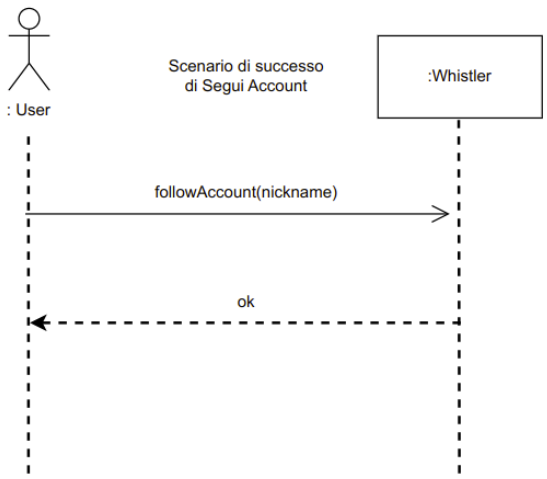
Diagramma di sequenza di sistema (SSD)

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).

Analizzando lo scenario principale di successo del caso d'uso **UC1** si è ottenuto il seguente SSD:



mentre per ciò che concerne lo scenario principale di successo del caso d'uso **UC2** si è ottenuto il seguente SSD:



Si è deciso di verificare la presenza dell'account, associato al nickname fornito, e la sua conseguente individuazione direttamente all'interno del sistema.

Contratti delle Operazioni

Di seguito si riportano i contratti delle operazioni di sistema 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: signUpAccount
Operazione :
signUpAccount(nickname:String,name:String,surname:String,email:String, password:String)
Riferimenti : caso d'uso: Gestisci Account
Pre-condizioni : nessuna
Post-condizioni :
- è stata creata una nuova istanza a di Account;
- gli attributi di a sono stati inizializzati;
- Whistler è stato associato all'account a tramite l'associazione "Registra";

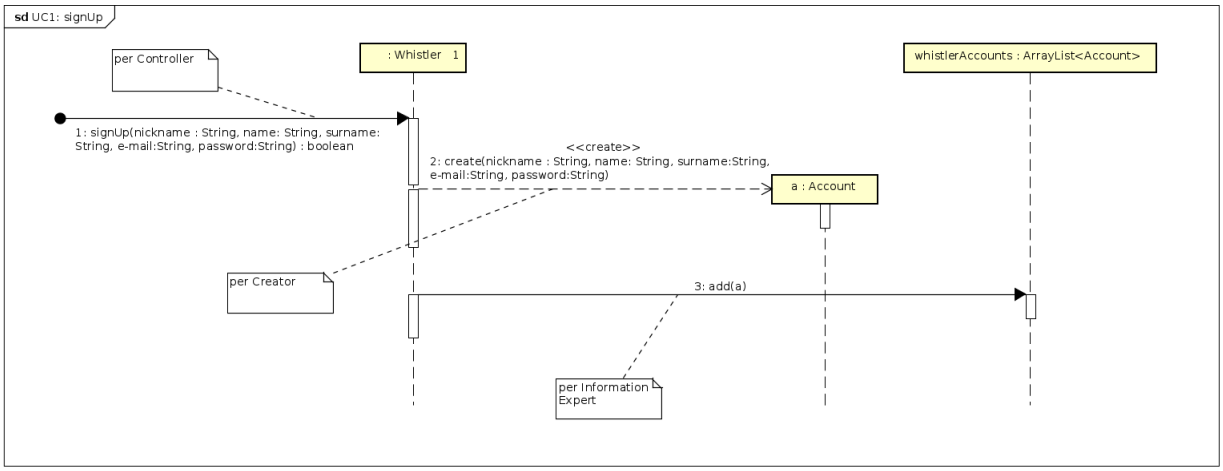
Contratto CO2: followAccount
Operazione : followAccount(nickname:String)
Riferimenti : caso d'uso: Segui Account
Pre-condizioni :
- l'Utente possiede un account ed è autenticato nella piattaforma;
- è presente almeno un altro account nella piattaforma oltre a quello dell'Utente;
Post-condizioni :
- l'account a è stato associato all'account identificato dal nickname fornito tramite l'associazione "Segue";

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)**.

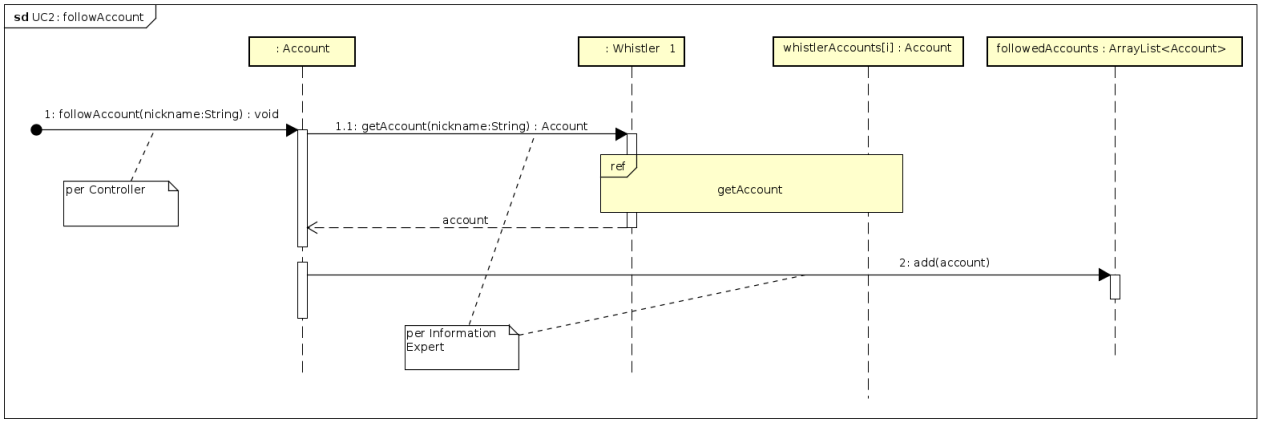
Diagrammi di Sequenza

- Creazione di un account su Whistler

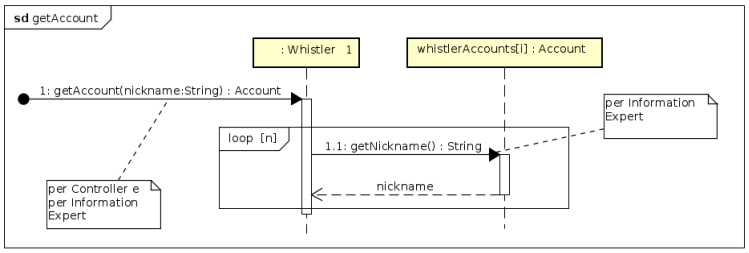


E' stato utilizzato il **Pattern Singleton (GoF)** sull'oggetto Whistler perché è necessario che esista un'unica istanza della piattaforma di microblogging.

- Seguire un account



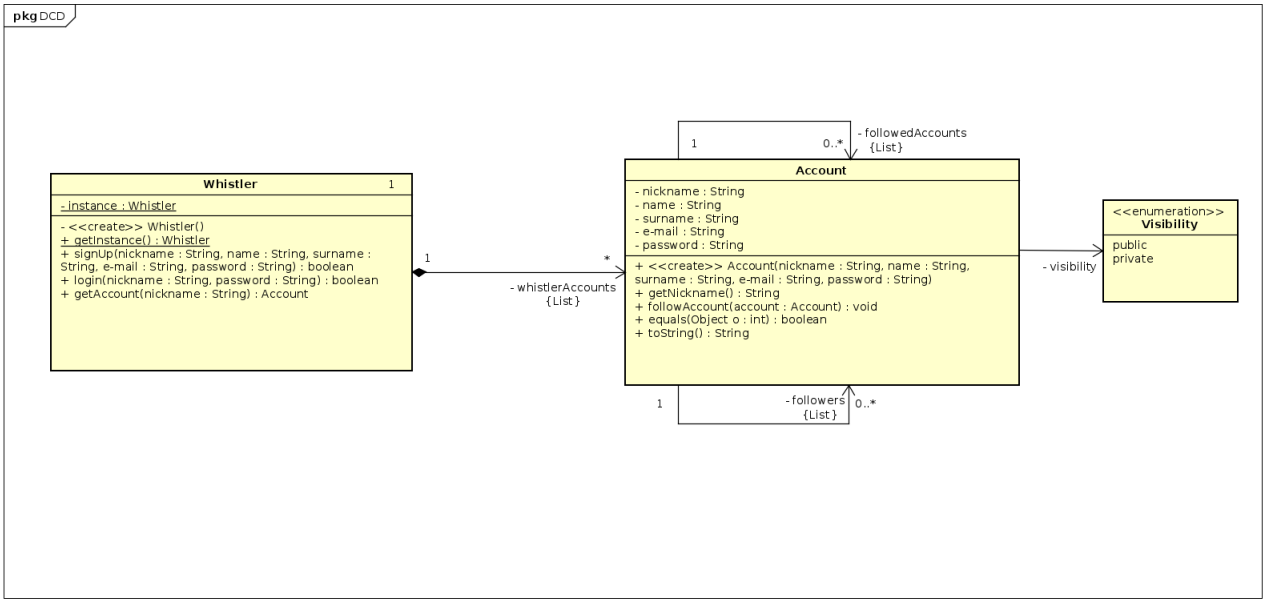
- getAccount:



[NOTA] :
Procedendo con la seconda iterazione mi sono accorto di aver dimenticato di considerare la creazione dell'associazione "E'-seguito" tra l'account a e l'account del whistleblower che si vuole seguire e di aver considerato unicamente l'associazione "Segue".
Si rimanda per la versione rivista e corretta del contratto delle operazioni e del diagramma di sequenza "UC2: followAccount" alla seconda elaborazione, o all'immagine "revised_DS_UC2_followAccount.png" nella cartella "01_Elaborazione_1".

Diagramma delle Classi

- DCD - Elaborazione 1



Testing

Lo sviluppo guidato dai test (**TDD** - Test-Driven Development) è un ottima pratica applicabile su UP. In generale il TDD prevede l'utilizzo di diversi tipi di test (**test unitari**, **test di Integrazione**, **test di Sistema**, **test di Accettazione**, **test di Regressione**).

I **test unitari** permettono di verificare il funzionamento delle piccole parti ("unità") del sistema, senza però verificare il sistema nel suo complesso, che vengono verificate invece, per mezzo dei **test di Sistema**. Esempi di unità sono: una classe oppure un metodo.

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.

Classi di test:
Whistler Class

- **SignUp Tests**
  - 1) testSignUp_ValidNicknameAndPassword **(CE)**, **(VE)**,**(CF)**
  - 2) testSignUp_NicknameAlreadyExists **(CF)**
  - 3) testSignUp_NicknameWithSpaces **(CE)**
  - 4) testSignUp_PasswordLengthEqualEight **(VE)**
  - 5) testSignUp_PasswordLengthLessThanEight **(VE)**

- **Login Tests**
  - 1) testLogin_InvalidNicknameAndPassword **(CF)**
  - 2) testLogin_InvalidNickname **(CF)**
  - 3) testLogin_InvalidPasswordLength **(VE)**, **(CF)**
  - 4) testLogin_ValidNicknameAndPasswordLenghtButNotValidPassword **(CF)**,**(VE)**
  - 5) testLogin_ValidNicknameAndPassword **(CF)**, **(VE)**

- **SearchAccount Tests**
  - 1) testSearchAccount_IsPresent **(CE)**, **(CF)**
  - 2) testSearchAccount_NotPresent **(CE)**, **(CF)**

Account Class
- 1) testFollowAccount_TwoAccountInWhistler **(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** scelti sono basati sulla:
- **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 al fine di rendere più robusta la gestione degli input inseriti dall'utente, durante l'interazione con il Sistema, e di migliorare il comportamento dei metodi presi in esame. In particolare sono stati gestiti i seguenti casi:

- (Login) L'inserimento di nickname non esistenti nel sistema
- (Login) L'inserimento di password errata
- (SignUp) L'inserimento di un nickname già esistente

- (Signup/Login) L'inserimento di nickname contenenti spazi
- (SignUp/Login) L'inserimento di una password contenente un numero di caratteri inferiore ad 8
- (In tutte le console) L'inserimento di comandi non validi o fuori range

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.

Iterazione 1 - Refactoring

1 Introduzione

Dopo aver completato di implementare il codice relativo alla prima iterazione, approcciandomi ai requisiti della seconda iterazione, mi sono reso conto che sarebbe stato necessario aggiungere uno strato di persistenza al progetto prima di poter proseguire. In questo modo l'implementazione dei requisiti, previsti dalla seconda iterazione, sarebbe diventata più semplice da affrontare.
Un'applicazione di microblogging vive degli Utenti e dei Post inseriti da quest'ultimi, l'idea di dover inserire gli utenti ed i relativi post, manualmente o tramite metodo di inizializzazione, ogni volta che la piattaforma viene spenta e riaccesa non mi entusiasma.

Lo scopo di questa prima iterazione di Refactoring, dunque, è quello di soddisfare i seguenti requisiti:

- Implementazione di uno strato di persistenza
- Consistenza dei dati

Gli elaborati del **Modello di Dominio** e dei **Sequence System Diagrams - SSD** sono rimasti invariati, rispetto a quelli riportati nella documentazione della prima iterazione.

Al fine di isolare maggiormente il livello **applicativo/di business** da quello di **persistenza** si è deciso di applicare il pattern **Data Access Object - (DAO)**. Quest'ultimo consiste nel realizzare delle classi che incapsulano l'interazione con il database.
In questo caso, però, si è deciso di interagire con il database sfruttando le API offerte dall'ORM framework **Hibernate**, al fine di implementare persistenza e transazionalità in modo rapido e semplificare lo sviluppo. La scelta di inserire tali API in opportune classi DAO consente non solo di "nascondere ulteriormente" la complessità di interazione con il database sottostante, permettendo ai due livelli di evolvere saparatamente senza che l'uno conosca i dettagli dell'altro, ma evita anche la necessità di doversi ricordare di inserire ogni volta operazioni a corredo di una transazione.

Nelle classi DAO, infatti, sono stati realizzati dei metodi (CRUD) che gestiscono, di volta in volta, l'apertura e la chiusare della **sessione e le transazioni Hibernate** necessarie per l'interazione con il database.

Il mapping tra gli oggetti Java, le entities e le relazioni del database scelto (**MySql**) è stato realizzato per mezzo di descrittori XML.

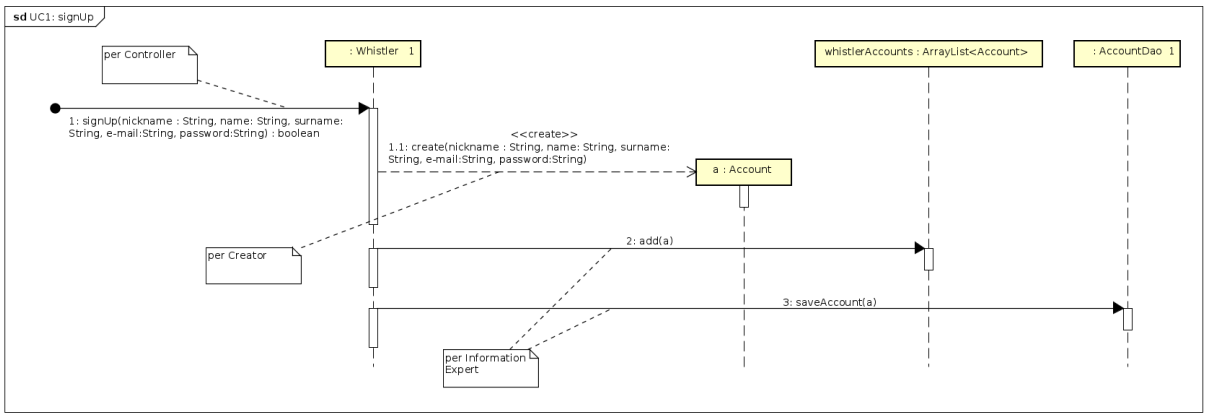
Per questa iterazione è stata realizzata la classe **AccountDao.java** ed il file **Account.hbm.xml** necessario per il mapping object/relational messo a disposizione da Hibernate, il quale è un'implementazione standard della **specifica JPA - Java Persistence API**. Il focus di JPA è quello di occuparsi della persistenza delle classi **POJO - Plain Old Java Object**, ovvero delle classi Java non strettamente legate ad uno specifico framework.

Si prevede di proseguire con la realizzazione delle classi DAO, anche nelle iterazioni successive, per ciascuna classe POJO da rendere permanente.

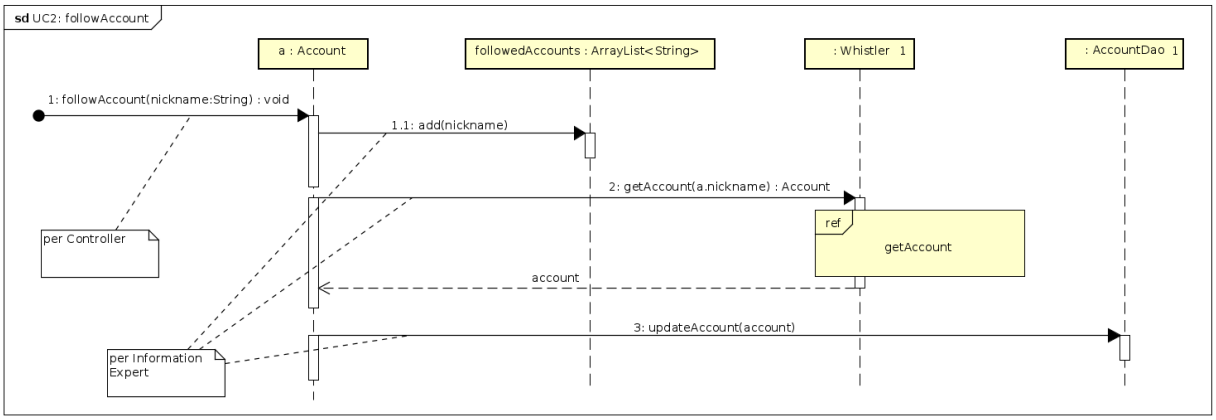
Di seguito si riportano i Diagrammi di Interazione (nello specifico **Diagrammi di Sequenza**) - modellazione dinamica - ed il **Diagramma delle classi** - modellazione statica - rivisti in seguito alle modifiche apportate.

Diagrammi di Sequenza

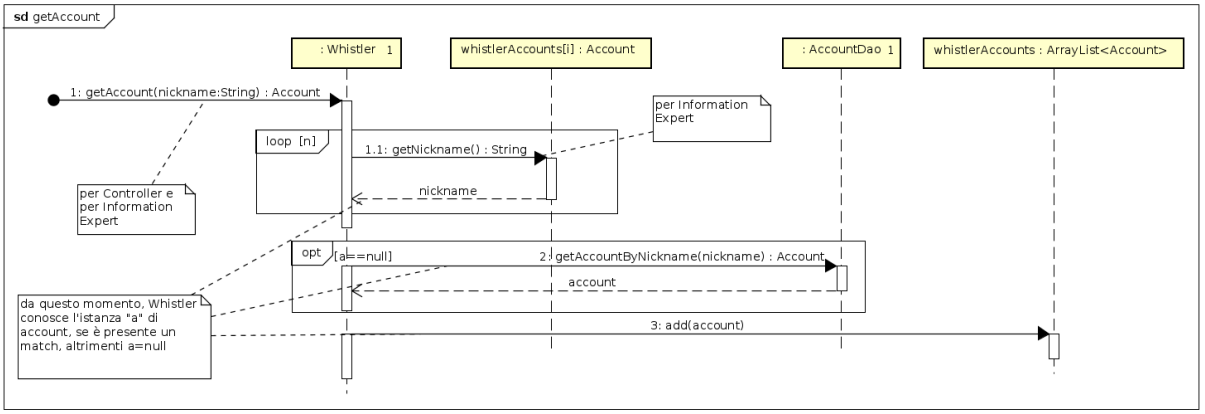
- Creazione di un account su Whistler



- Seguire un Account:



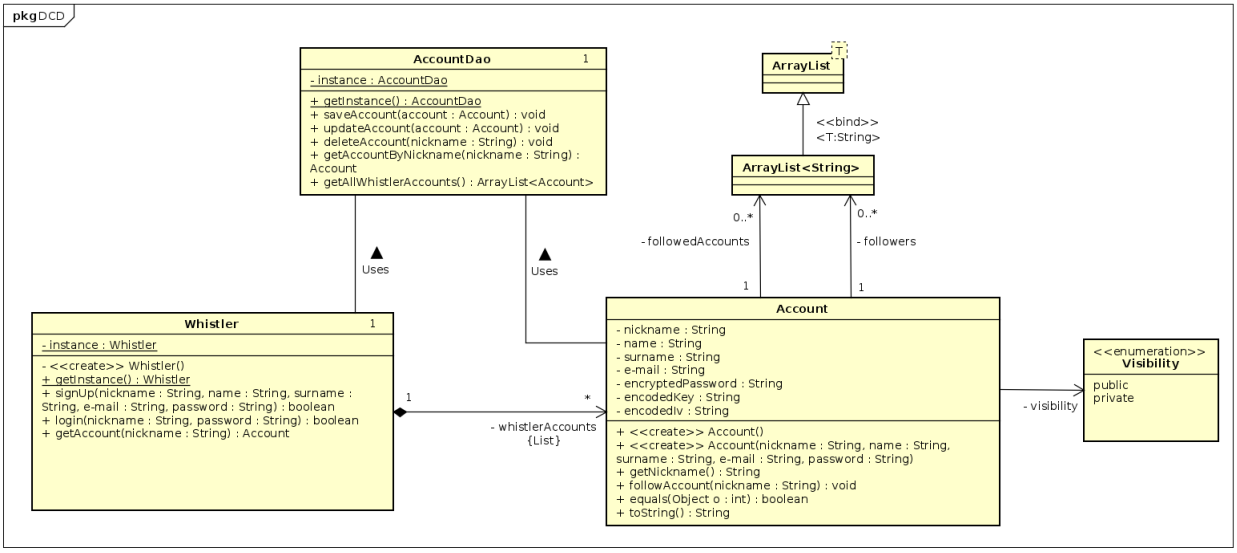
- getAccount:



[NOTA] :
Procedendo con la seconda iterazione mi sono accorto di aver dimenticato di considerare la creazione dell'associazione "E'-seguito" tra l'account a e l'account del whistleblower che si vuole seguire e di aver considerato unicamente l'associazione "Segue".
Si rimanda per la versione rivista e corretta del diagramma di sequenza "UC2: followAccount" alla seconda elaborazione, o all'immagine "revised_DS_UC2_followAccount.png" nella cartella "02_Refactoring_Elaborazione_1".

Diagramma delle Classi

- DCD - Elaborazione 1 - Refactor



Durante il refactoring si è deciso che fosse più opportuno rendere **followedAccounts** e **followers** del tipo ArrayList di Stringhe, contenente i nicknames degli account piuttosto che gli Account per intero, in modo da snellire le informazioni associate al singolo Account.

Per le successive iterazioni, per evitare che il diagramma delle classi risulti eccessivamente confusionario, verrà omessa la rappresentazione delle classi DAO associate a ciascuna classe persistente, essendo la modalità di utilizzo affine a quella appena mostrata. Per quanto riguarda, invece, i diagrammi di sequenza, le classi DAO verranno mostrare solamente quando necessarie per la comprensione dell'implementazione dell'operazione, in tutti gli altri casi verranno omesse.

Testing

Test Unitario

Le classi di test sono rimaste pressocché invariate rispetto a quelle individuate nell'iterazione precedente, al di là di qualche modifica di implementazione dovuta alla presenza dello strato di persistenza.

Test di Sistema

Si è scelto di eseguire dei test di sistema manuali, provando a portare a termine i requisiti funzionali indicati nella prima iterazione, riavviando il sistema al fine di testare la persistenza dei dati inseriti.

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.

Elaborazione - Iterazione 2

1 Introduzione

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

1. Gli utenti possono pubblicare dei **post** costituiti da un titolo, dal contenuto e da parole chiave.
2. L'utente, oltre a visualizzare nella propria **home** i post dei whistleblower seguiti, può visualizzare post pubblici di qualsiasi whistleblower nella bacheca.
3. Il **profilo** contiene i post pubblicati dall'utente e le sue informazioni personali, che possono essere modificate e delle quali è possibile definire la visibilità (pubblica e quindi visibile agli altri utenti o privata).
4. L'utente può visualizzare e gestire gli utenti presenti nella sua cerchia di interesse.
5. La piattaforma tiene traccia di tutte le operazioni effettuate registrando le azioni svolte dagli utenti e memorizzandole in un opportuno log.

I casi d'uso di riferimento sono **UC3: Gestisci Post**, **UC6: Visualizza Home** e **UC8: Visualizza Profilo**.

Nello specifico ci si concentra sulla:

- implementazione dello scenario principale di successo per ciò che riguarda il caso d'uso **UC3**. Inoltre si prendono in considerazione anche le estensioni **1a** ed **1b**. Nell'ordine: "inserimento", "modifica" e "rimozione" di un post. In questa iterazione verrà tralasciata l'estensione **5a** relativa alla verifica del numero di caratteri consentiti in un post.
- implementazione del caso d'uso **UC6**, ovvero la visualizzazione della home, contenente i post degli utenti presenti nella cerchia d'interesse dell'account dell'utente.
- implementazione dello scenario principale di successo e degli scenari alternativi del caso d'uso **UC8**, soffermandosi sull'estensione **1a** (visualizzazione del proprio profilo)

- registrazione delle operazioni effettuate nel sistema su un file log.

Inoltre, verranno implementati anche gli scenari tralasciati durante la prima iterazione, relativi ai casi d'uso **UC1: Gestisci Account** e **UC2: Segui Account** in modo da coprirli nella loro interezza.

- implementazione degli scenari alternativi del caso d'uso **UC1**, messi da parte nell'iterazione 1, ovvero la modifica (**1a**) e la rimozione (**1b**) dell'account.
- implementazione dello scenario alternativo, non ancora implementato, del caso d'uso **UC2** relativo alla gestione degli account seguiti presenti nella cerchia d'interesse dell'utente.

Si rimandano a successive iterazioni, l'implementazione della **bacheca** in cui l'utente può visualizzare i post pubblici dei whistleblowers non presenti nella sua cerchia d'interesse e l'invio di una notifica verso i follower dell'utente, alla pubblicazione di un nuovo post.

Sono stati riscritti i casi d'uso **UC6** e **UC8** 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.

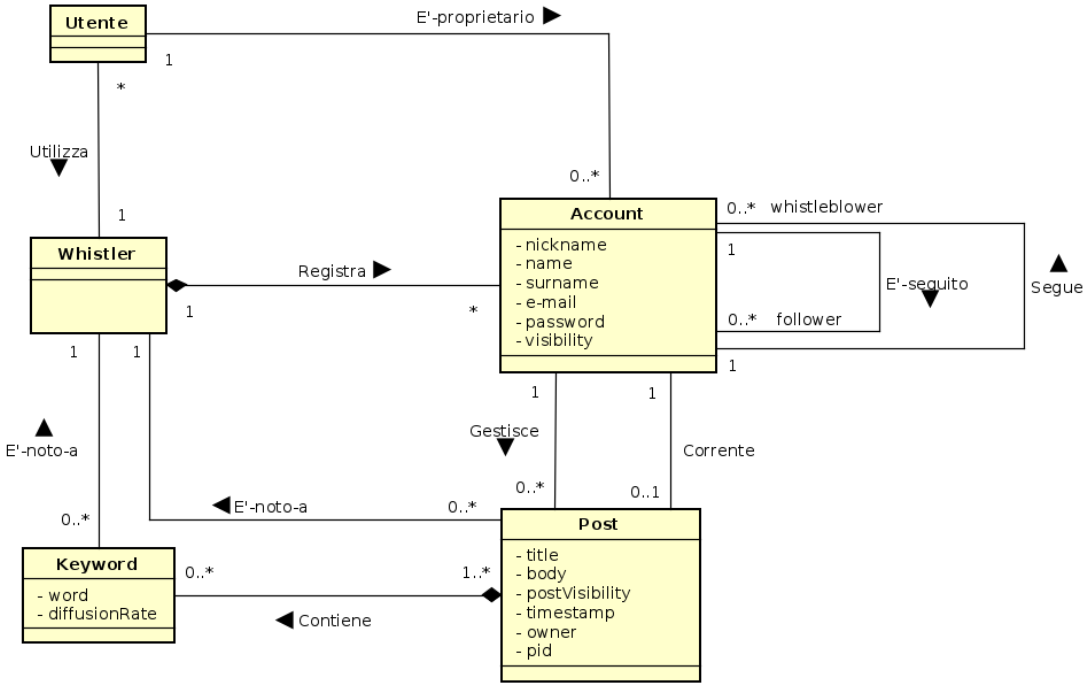
Modello di Dominio

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 esame il caso d'uso **UC3** è possibile identificare le seguenti classi concettuali:

- **Post:** è il messaggio inserito all'interno della piattaforma di microblogging
- **Keyword:** è la parola che identifica al meglio il significato del post nel quale è inserita. E' costituita dalla parola in sé e dal tasso di diffusione (utile per individuare le keywords di tendenza).

Il modello di dominio ricavato è il seguente:



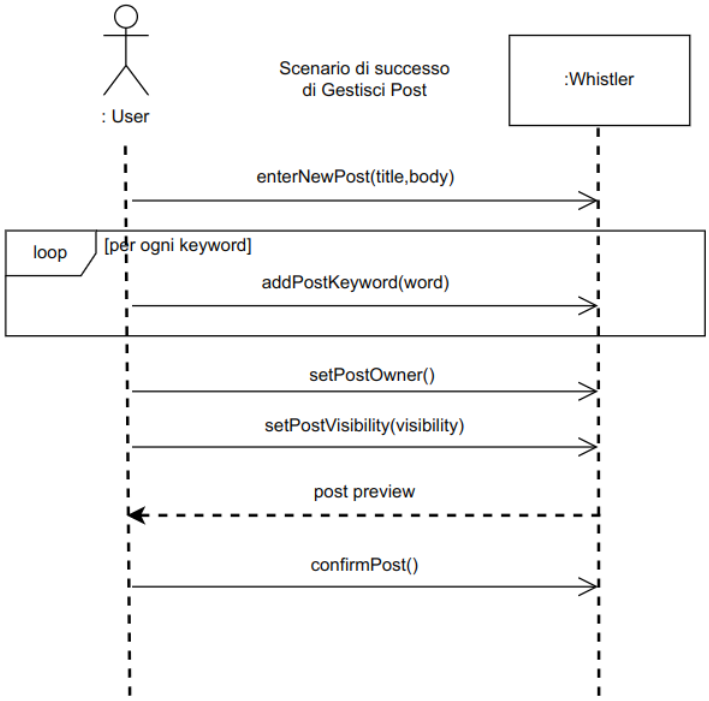
Prendendo in esame, invece, i casi d'uso **UC6** e **UC8** non emergono particolari classi concettuali. Sebbene la timeline possa sembrare un buon candidato, si sceglie di rilegare le dinamiche di aggiornamento della visualizzazione alla user interface, per cui il modello di dominio rimane invariato.

Anche le estensioni dei casi d'uso **UC1** e **UC2** non comportano variazioni nel modello di dominio fin qui ottenuto.

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.

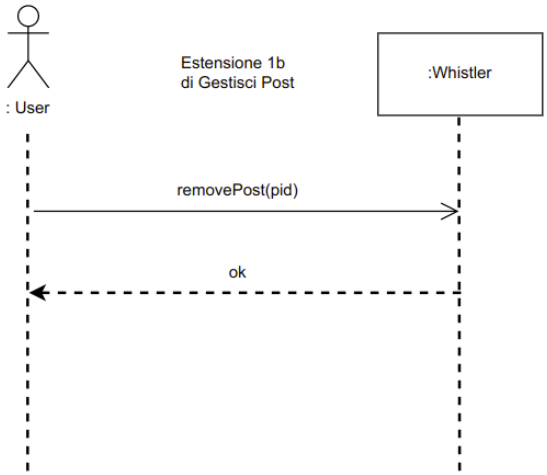
Per ciò che riguarda il caso d'uso **UC3**:



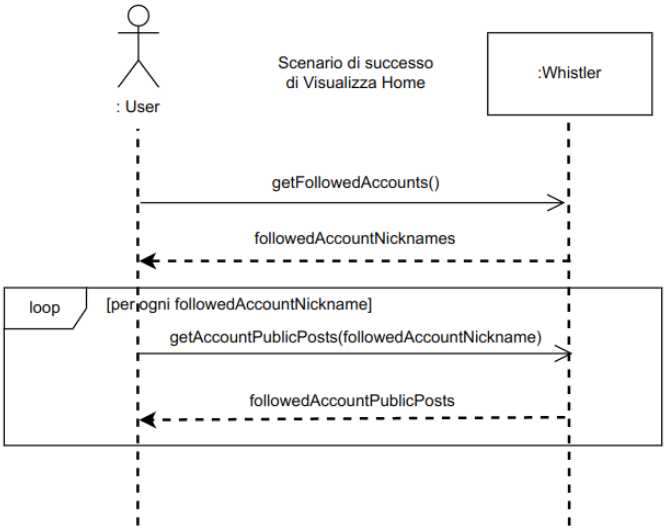
Durante l'inserimento di ciascun post: l'operazione d'inserimento del **timestamp** e della generazione del **pid** (post_id) vengono gestite internamente dal sistema.

Scenari alternativi 1a e 1b:
Affinché si possa modificare o eliminare un post è necessario che l'utente richiedente ne sia il proprietario, altrimenti non sarà possibile la modifica o la rimozione. Il controllo in questione viene eseguito direttamente all'interno del sistema.
Relativamente alle operazioni di modifica(**1a**), poiché queste, di volta in volta, si limitano a singole chiamate del tipo "setTitle", "setBody" ecc... in funzione di ciò che l'utente vuole modificare del post, si è deciso di omettere i relativi SSD.

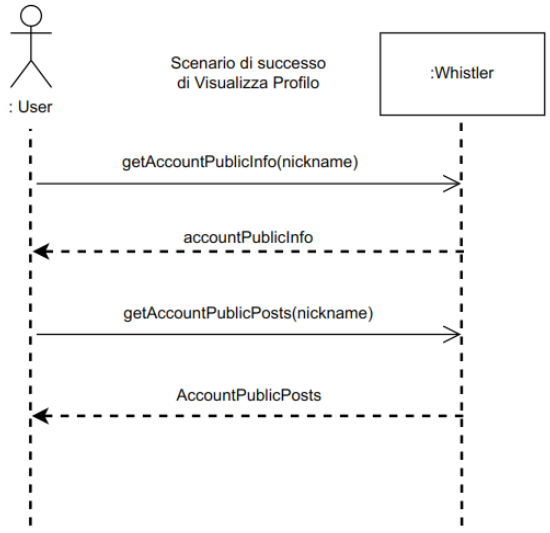
Si riporta di seguito, invece, l'SSD dell'estensione **1b** relativa alla rimozione di un post.



Per il caso d'uso **UC6**:

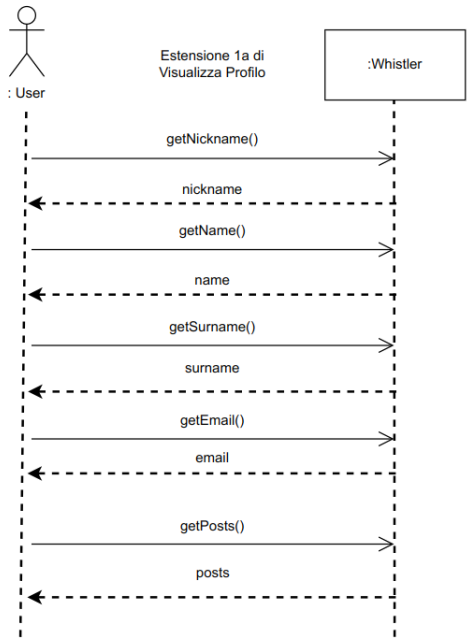


Per il caso d'uso **UC8**:



Tale caso d'uso è stato analizzato adottando come precondizione la presenza, nella piattaforma, dell'account associato al nickname in esame.

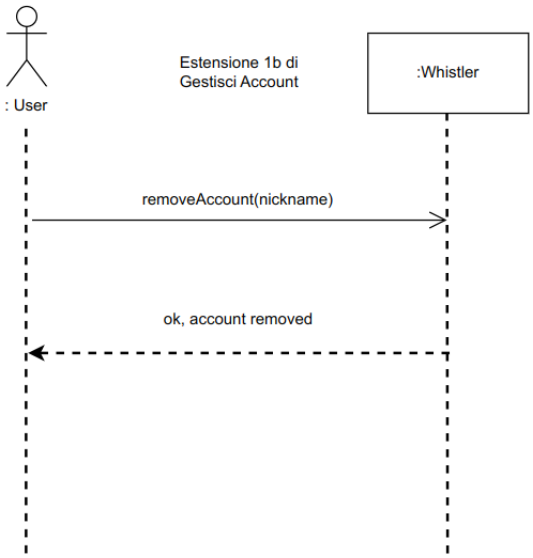
Relativamente all'estensione **1a**:



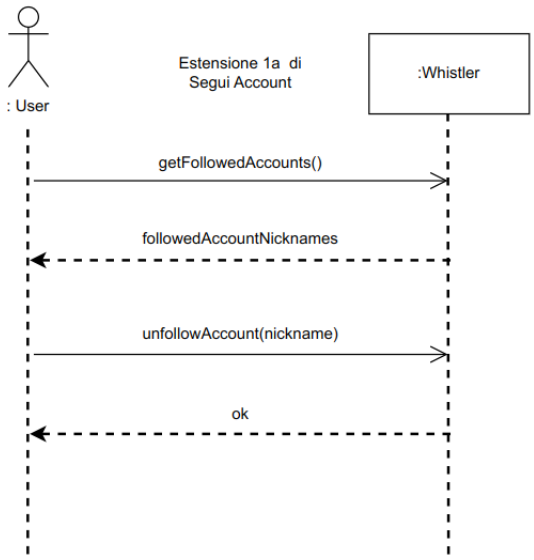
Si noti come l'SSD dell'estensione **1a** di **UC8** rimanga pressoché invariato nel risultato ottenuto rispetto allo scenario principale di successo. La differenza principale consiste nel ritornare tutte le informazioni ed i post (pubblici e privati) dell'account e non solo i post e le informazioni pubbliche.

Analizzando le estensioni dei casi d'uso **UC1** ed **UC2** si è ritenuto importante mostrare i seguenti SSD:

Estensione **UC1 1b**:



Estensione **UC2 1a**:



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: enterNewPost
Operazione : enterNewPost(title:String, body:String)
Riferimenti : caso d'uso: Gestisci Post
Pre-condizioni :
- l'Utente possiede un account ed è autenticato nella piattaforma
Post-condizioni :
- è stata creata una nuova istanza p di Post;
- gli attributi di p sono stati inizializzati;
- p è stato associato ad Account tramite l'associazione "Corrente"

Contratto CO2: addPostKeyword
Operazione : addPostKeyword(word:String)
Riferimenti : caso d'uso: Gestisci Post
Pre-condizioni :
- è in corso l'inserimento di un Post p
Post-condizioni :
- è stata creata una nuova istanza k di Keyword;
- gli attributi di k sono stati inizializzati;
- la keyword k è stata associata al Post corrente p tramite l'associazione "Contiene"

Contratto CO3: setPostVisibility
Operazione : setPostVisibility(vis:Visibility)
Riferimenti : caso d'uso: Gestisci Post
Pre-condizioni :
- è in corso l'inserimento di un nuovo Post p
Post-condizioni :
- p.visibility è diventata vis.

Contratto CO4: confirmPost
Operazione : confirmPost()
Riferimenti : caso d'uso: Gestisci Post
Pre-condizioni :
- è in corso l'inserimento di un nuovo Post p
Post-condizioni :
- è stata associata l'istanza p di Post corrente ad Account tramite l'associazione "Gestisce".
- l'istanza p di Post corrente e le Keywords ad esso associate sono note a Whistler.

Contratto CO5: removePost
Operazione : removePost(pid:String)
Riferimenti : caso d'uso: Gestisci Post
Pre-condizioni :
- è presente un post p con pid corrispondente a quello in esame.
- l'account che si accinge a rimuovere il post è proprietario di quel post.
Post-condizioni :
- il diffusionRate delle keywords associate al post è stato ridotto;
- le associazioni di p con le keywords che conteneva sono state rimosse;
- l'associazione di p con Account è stata rimossa;
- p è stato rimosso.

Contratto CO6: removeAccount
Operazione : removeAccount(nickname:String)
Riferimenti : caso d'uso: Gestisci Account
Pre-condizioni :
- l'Utente possiede un account "a" ed è autenticato nella piattaforma
Post-condizioni :
- le associazioni del tipo "Gestisce" tra l'account a associato al nickname e ciascun post, di cui a è proprietario, sono state rimosse;
- i post di cui a è proprietario sono stati rimossi;
- le associazioni "Segue" ed "E'-seguito" tra l'account a ed altri account presenti nella piattaforma sono state rimosse;
- l'associazione "E'-proprietario" tra Utente ed Account è stata rimossa;
- l'account a è stato rimosso dalla piattaforma Whistler.

Contratto CO7: unfollowAccount
Operazione : unfollowAccount(nickname:String)
Riferimenti : caso d'uso: Segui Account
Pre-condizioni :
- l'account b associato al nickname fornito è presente nella cerchia d'interesse dell'account a dell'Utente
Post-condizioni :
- l'associazione "Segue" tra l'account a e l'account b è stata rimossa;
- l'associazione "E'-seguito" tra l'account b e l'account a è stata rimossa.

3 Errata corrige

Durante la stesura del contratto **unfollowAccount - C07** dell'elaborazione corrente, mi sono reso conto di non avere opportunamente sottolineato nel contratto **followAccount** dell'**elaborazione 1** la creazione dell'associazione "E'-seguito" tra l'account a e l'account b.

Si riporta di seguito la versione rivista e corretta:

Contratto CO2: followAccount - Elaborazione 1
Operazione : followAccount(nickname:String)
Riferimenti : caso d'uso: Segui Account
Pre-condizioni :
- l'Utente possiede un account "a" ed è autenticato nella piattaforma;
- è presente almeno un altro account nella piattaforma oltre a quello dell'Utente;
Post-condizioni :
- l'account a è stato associato all'account b identificato dal nickname fornito tramite l'associazione "Segue";
- *l'account b è stato associato all'account a tramite l'associazione "E'-seguito".*

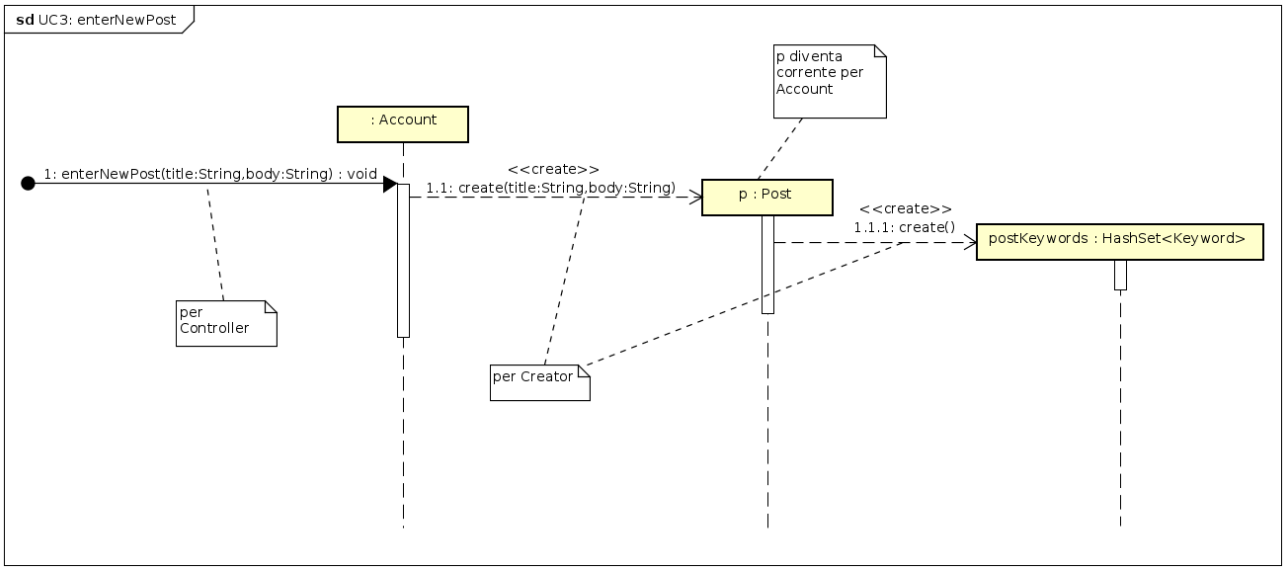
Questa dimenticanza si è propagata anche nella stesura del diagramma di sequenza **sd UC2: followAccount** - Elaborazione 1 e nel refactoring dell'Elaborazione 1, in cui si è aggiunto lo strato di persistenza. Per tale motivo nella sezione di progettazione verrà riporta anche la versione rivista e corretta del corrispettivo diagramma di sequenza.

4 Progettazione

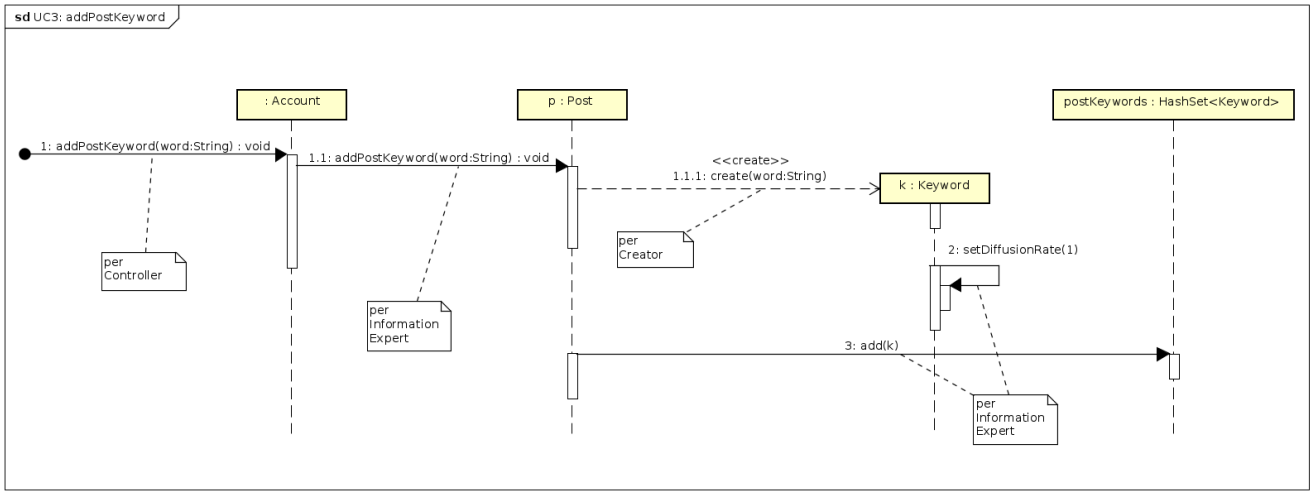
Dopo aver terminato l'analisi orientata ad oggetti per questa iterazione, sfruttando gli elaborati prodotti, si passa alla fase di progettazione. Di seguito si riportano i Diagrammi di Interazione (nello specifico **Diagrammi di Sequenza**) - modellazione dinamica - ed il **Diagramma delle classi** - 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)**.

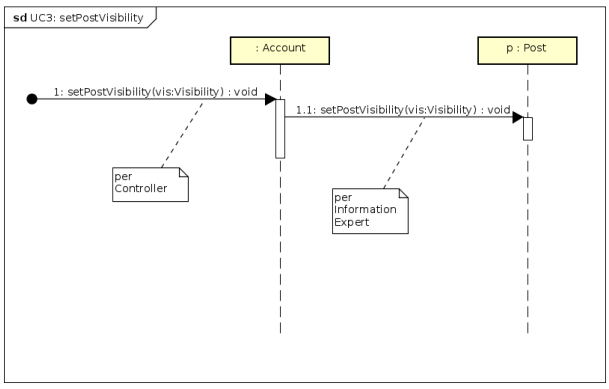
Diagrammi di Sequenza

- Creazione di un nuovo post su Whistler

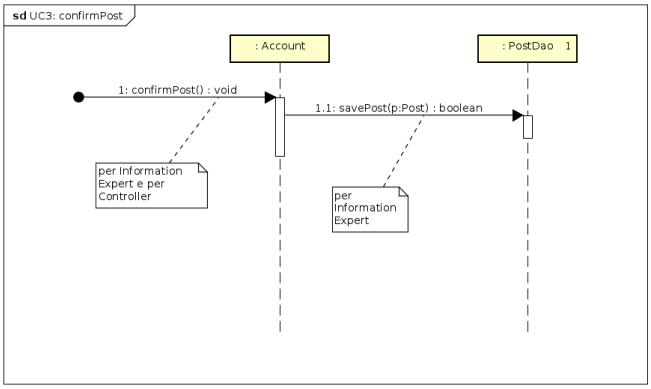


- Aggiunta di una nuova keyword al post corrente





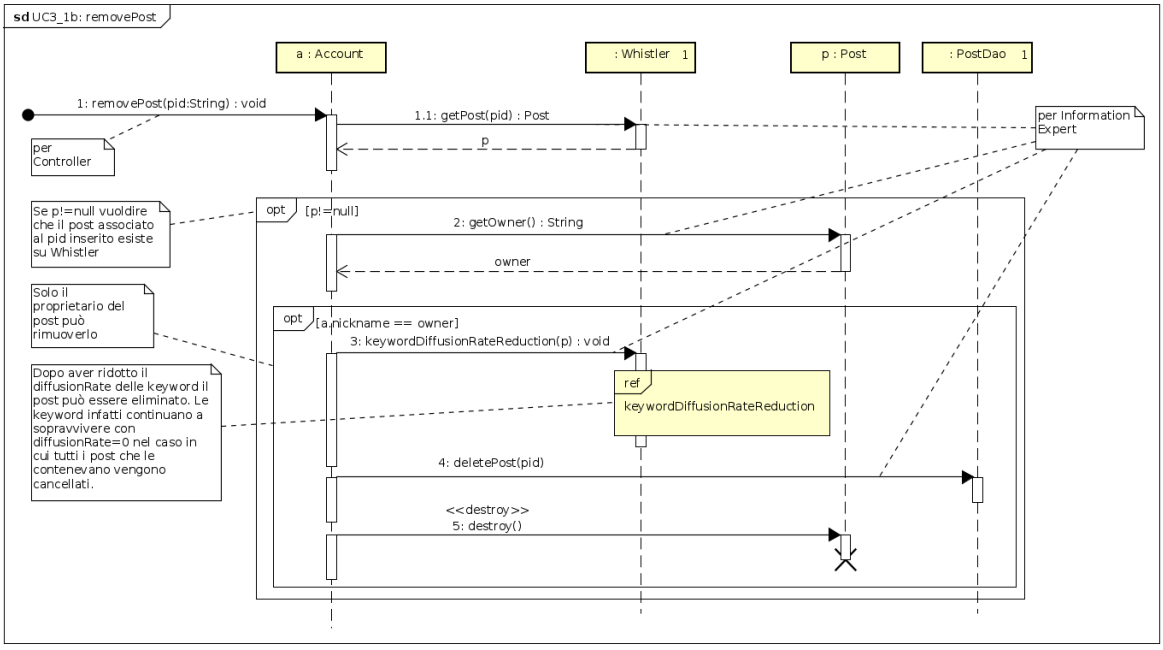
(a) Modifica della visibilità del post corrente

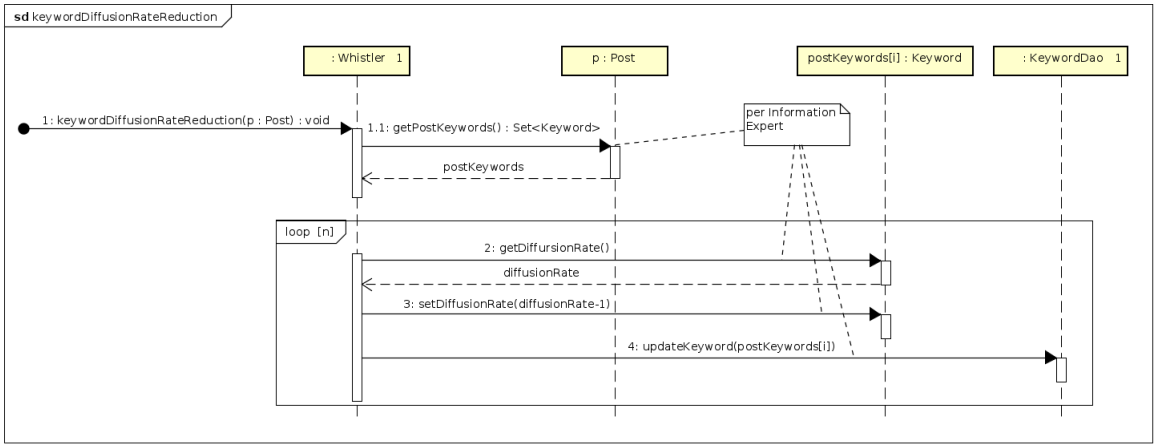


(b) Conferma creazione Post

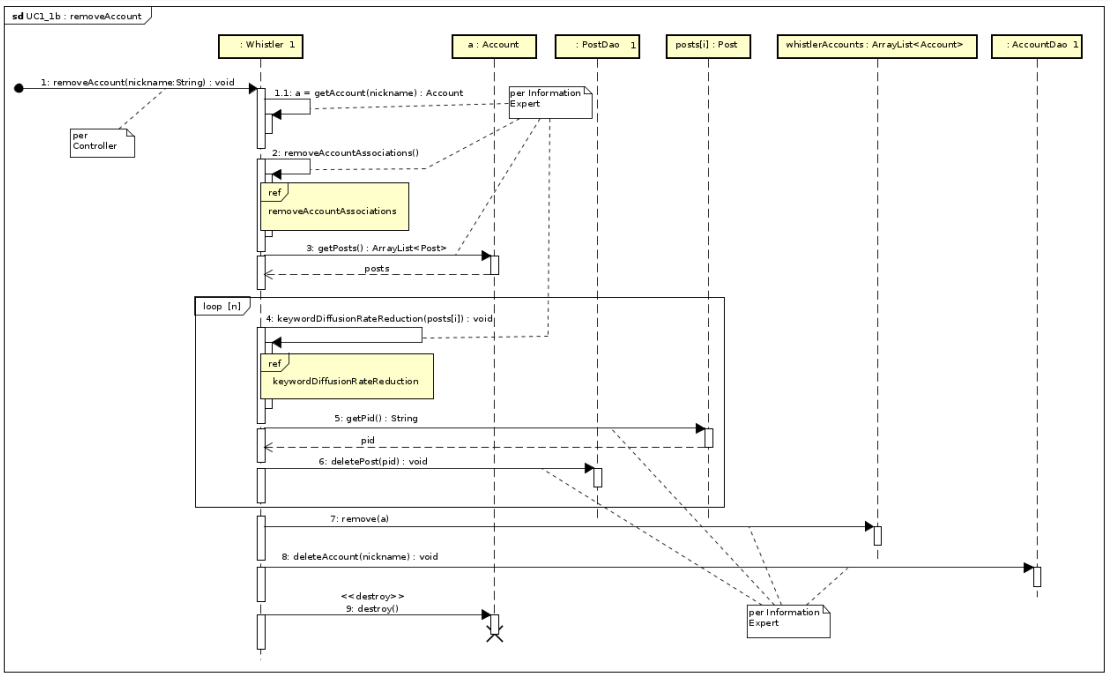
Si è deciso di gestire la conferma salvando il post direttamente in modo persistente, e successivamente riempire **posts: ArrayList<Post>** con i post presenti sul Database.

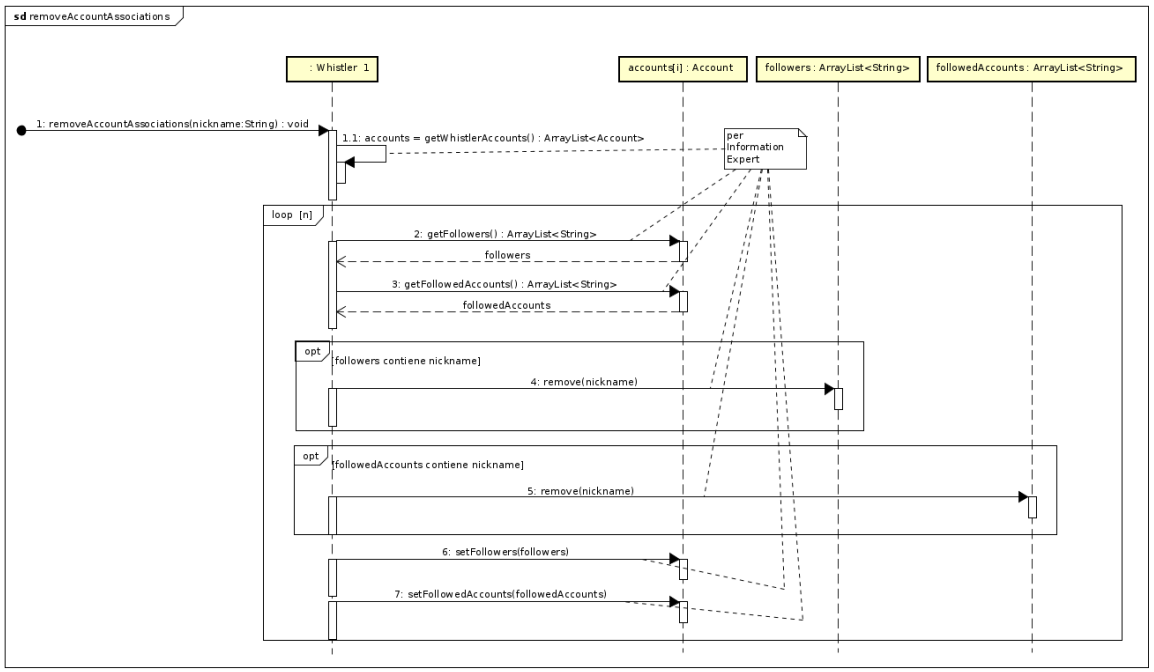
- Eliminazione di un Post



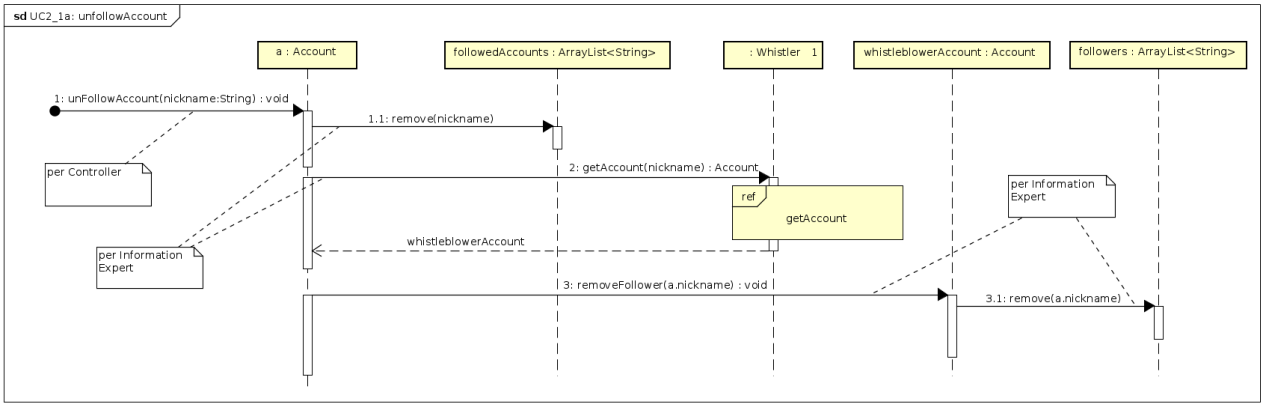


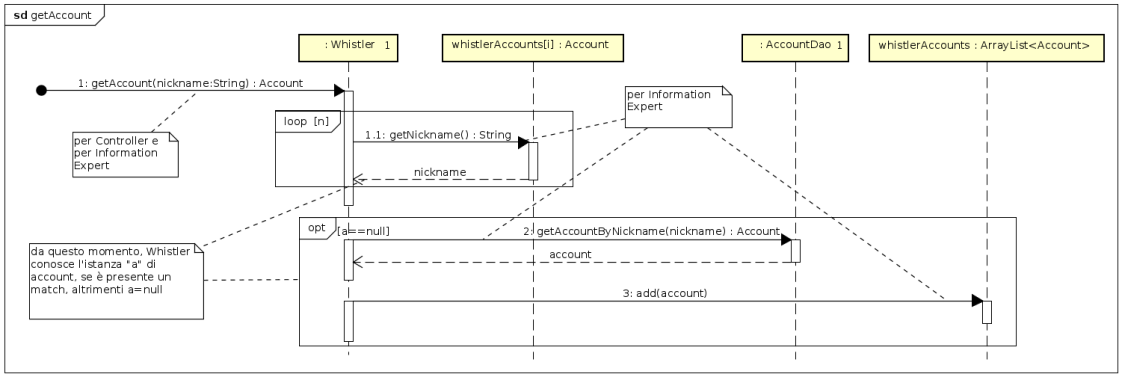
- Eliminazione dell'Account





- Rimozione dalla cerchia d'interesse di un Account

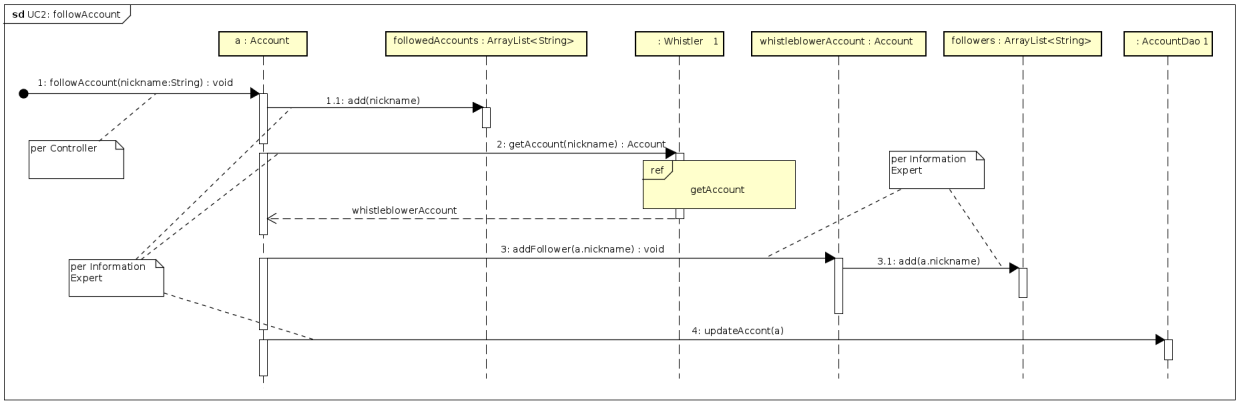




Errata Corrige - Diagrammi di sequenza - Elaborazione 1 e Refactoring

Si riporta di seguito il diagramma di sequenza **sd UC2: followAccount** rivisto e corretto aggiungendo anche la gestione dell'associazione "E'-seguito" da parte del whistleblower che l'account dell'Utente decide di seguire.
Si noti che il Diagramma di Sequenza mostra l'implementazione attuale, con **followedAccount** e **follower** di tipo ArrayList<**String**> e non ArrayList<**Account**> come era stato pensato inizialmente nella prima iterazione.

- Seguire un account



Diagrammi delle Classi

- DCD : Elaborazione 2
Si rimanda al file **Whistler_E2.asta** nella cartella 03_Elaborazione_2 per la consultazione del diagramma delle Classi relativo all'elaborazione 2.

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.

Classi di test:

Account Class
- 1) testUnFollowAccount **(CF)**
- 2) testConfirmPost **(CF)**
- 3) testRemovePost **(CF)**

Post Class
- 1) testAddPostKeyword **(CF)**

Whistler Class
- 1) testGetPost_IsPresent **(CE)(CF)**
- 2) testGetPost_NotPresent **(CE)(CF)**
- 3) testGetAccountPublicPosts **(CF)**
- 4) testGetAccountPublicInfo **(CF)**
- 5) testKeywordDiffusionRateReduction **(CF)**
- 6) testRemoveAccount **(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.

Elaborazione - Iterazione 3

1 Introduzione

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

1. Per ogni post l'utente può definirne la visibilità ed *il contenuto* ***non*** *può essere* ***superiore ai 280 caratteri****.*
2. La piattaforma offre la possibilità di ***ricercare i post*** di altri whistleblower sulla base di ***parole chiave****.*
3. E' possibile seguire gli account di altri whistleblower per visualizzare i post da loro pubblicati nella piattaforma, ***commentarli*** e mettere like.
4. L'utente, oltre a visualizzare nella propria home i post dei whistleblower seguiti, *può visualizzare post pubblici di qualsiasi whistleblower nella* ***bacheca****.*

I casi d'uso di riferimento sono **UC3: Gestisci Post**, **UC4: Ricerca Post**, **UC5: Gestisci Commento** e **UC7: Visualizza Bacheca**.

Nello specifico ci si concentra sulla:

- implementazione dell'estensione **5a** del caso d'uso **UC3**, relativa alla **verifica del numero di caratteri** consentiti in un post, tralasciata nell'iterazione precedente.
- implementazione dello scenario principale di successo del caso d'uso **UC4** e della sua estensione **4a**, ovvero la **ricerca di un post** contenente una determinata parola chiave e la gestione di una ricerca priva di risultato.
- implementazione dello scenario principale di successo per ciò che riguarda il caso d'uso **UC5**. Prendendo in considerazione anche le estensioni **1a** ed **1b**. Nell'ordine: "inserimento", "modifica" e "rimozione" di un **commento**.
- implementazione del caso d'uso **UC7**, ovvero la visualizzazione della **bacheca**, contenente i post pubblici di qualsiasi whistleblower, anche di coloro che non sono presenti nella cerchia d'interesse del singolo account.

Si rimandano a successive iterazioni:
- la gestione della notifica alla pubblicazione di un post
- l'inserimento del like ad un post

Sono stati riscritti i casi d'uso **UC4**, **UC5** e **UC7** 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 esame il caso d'uso **UC4** il modello di dominio rimane invariato rispetto a quello mostrato nell'Elaborazione 2, acquisiscono però maggior importanza le associazioni "E'-noto-a" tra Post e Whistler e tra Keyword e Whistler. Infatti, al fine di ricercare i post contenenti una determinata parola chiave Whistler deve conoscere i post presenti nella piattaforma e le keywords.

Lo stesso avviene per il caso d'uso **UC7** nello specifico per l'associazione "E'-nota-a" tra Keyword e Whistler. Infatti, al fine di poter definire quali sono le Keywords di tendenza, necessarie per la visualizzazione nella bacheca dei post pubblici che le contengono, il Sistema deve conoscere tutte le Keywords inserite nella piattaforma.

Prendendo in considerazione, invece, il caso d'uso **UC5** è possibile identificare la seguente classe concettuale:

- **Commento:** è il messaggio di risposta riferito al contenuto di un Post

Il modello di dominio ricavato è il seguente:

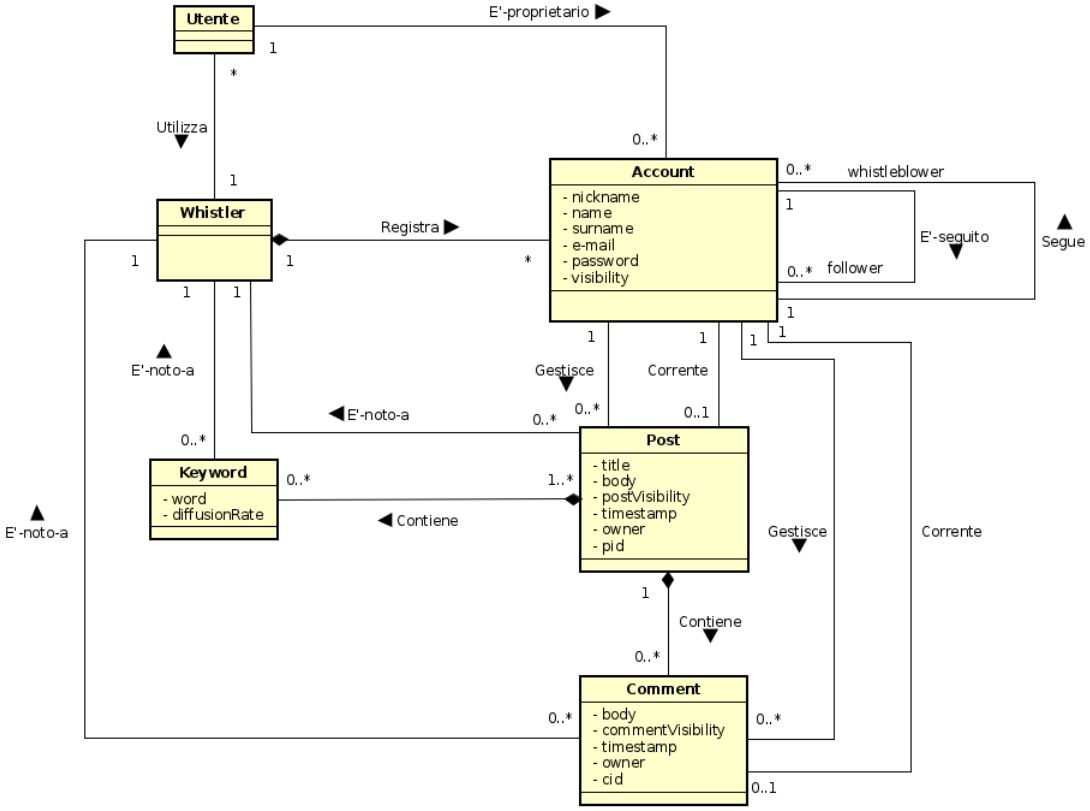
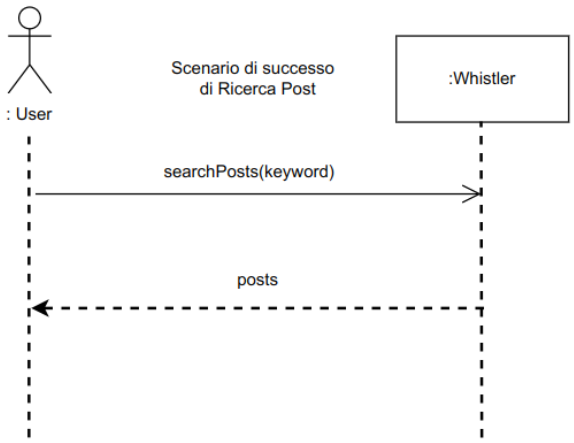


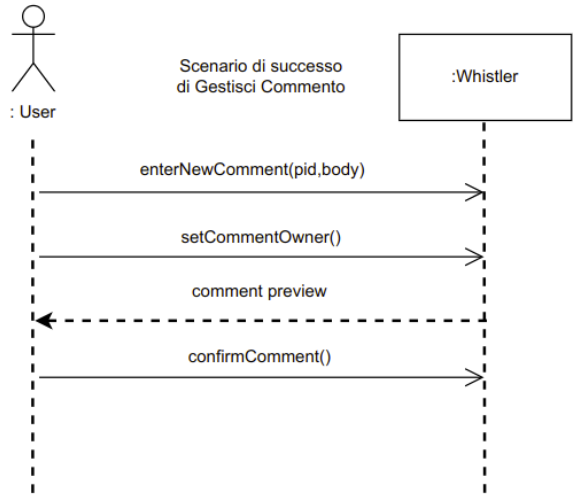
Diagramma di sequenza di sistema (SSD)

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 **UC4**:

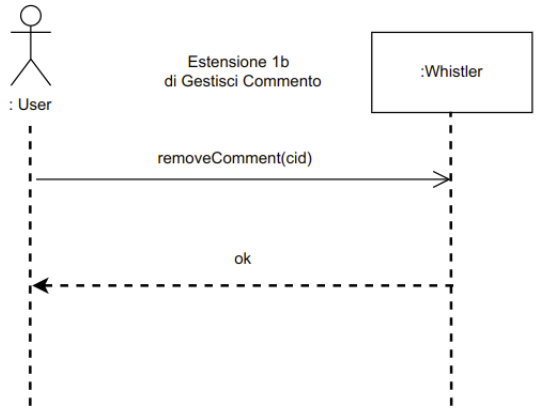


Per il caso d'uso **UC5**:

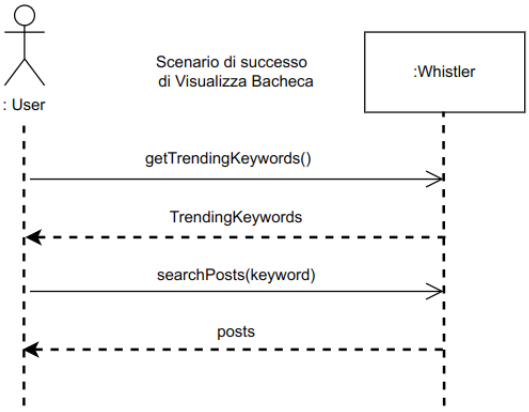


Durante l'inserimento di ciascun commento: l'operazione d'inserimento del **timestamp**, la generazione del **cid** (comment_id) e la **visibilità** del commento vengono gestite internamente dal sistema. Nello specifico la visibilità del commento sarà identica a quella del post a cui è riferito.

Si riportano di seguito l'SSD dell'estensione **1b** relativa alla rimozione di un commento e l'SSD dello scenario di successo del caso d'uso **UC7**.



(a) estensione **UC5_1b** relativa alla rimozione di un commento



(b) caso d'uso **UC7**

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: searchPosts
Operazione : searchPosts(keyword:String)
Riferimenti : caso d'uso: Ricerca Post
Pre-condizioni :
- l'Utente possiede un account ed è autenticato nella piattaforma
Post-condizioni :
- il Sistema ha ritornato i posts contenenti la keyword indicata

Contratto CO2: enterNewComment
Operazione : enterNewComment(pid:String, body:String)
Riferimenti : caso d'uso: Gestisci Commento
Pre-condizioni :
- l'Utente possiede un account ed è autenticato nella piattaforma
- è presenta almeno un post, visibile all'Utente, nella piattaforma
Post-condizioni :
- è stata creata una nuova istanza c di Commento;
- gli attributi di c sono stati inizializzati;
- c è stato associato ad Account tramite l'associazione "Corrente"

Contratto CO3: setCommentOwner
Operazione : setCommentOwner()
Riferimenti : caso d'uso: Gestisci Commento
Pre-condizioni :
- è in corso l'inserimento di un nuovo Commento c
Post-condizioni :
- c.owner è diventato account.nickname

Contratto CO4: confirmComment
Operazione : confirmComment()
Riferimenti : caso d'uso: Gestisci Commento
Pre-condizioni :
- è in corso l'inserimento di un nuovo Commento c
Post-condizioni :
- è stata associata l'istanza c di Commento corrente ad Account tramite l'associazione "Gestisce";
- è stata associata l'istanza c di Commento corrente al Post, avente il pid indicato, tramite l'associazione "Contiene";
- l'istanza c di Commento corrente è nota a Whistler.

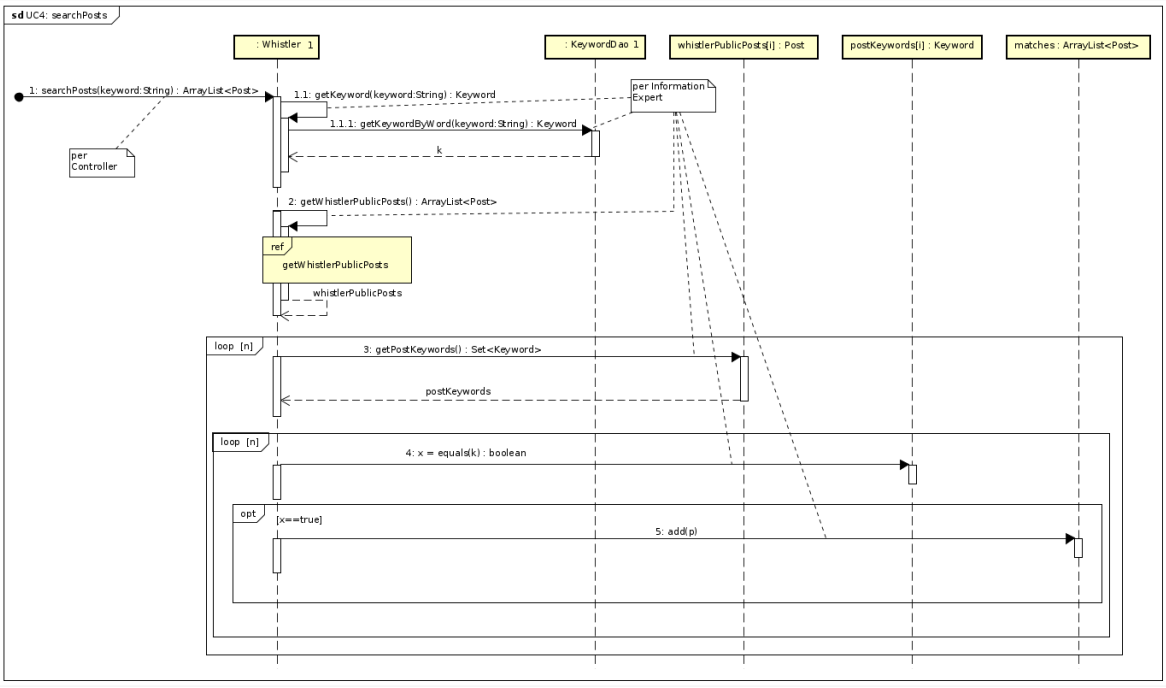
Contratto CO5: removeComment
Operazione : removeComment(cid:String)
Riferimenti : caso d'uso: Gestisci Commento
Pre-condizioni :
- su Whistler è presente un commento c con cid corrispondente a quello in esame;
- l'account che si accinge a rimuore il commento è proprietario di quel commento
Post-condizioni :
- l'associazione di c con il Post che lo conteneva è stata rimossa;
- l'associazione di c con Account è stata rimossa;
- c è stato rimosso.

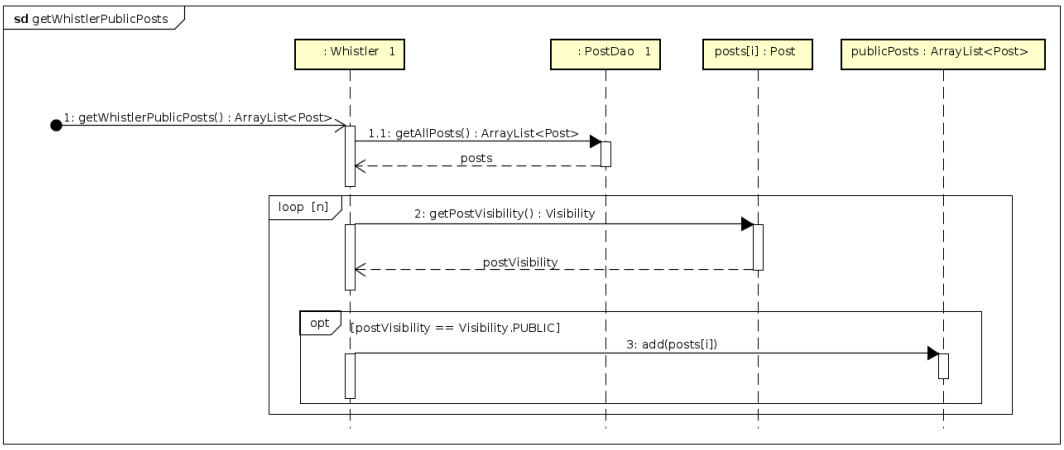
3 Progettazione

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)**.

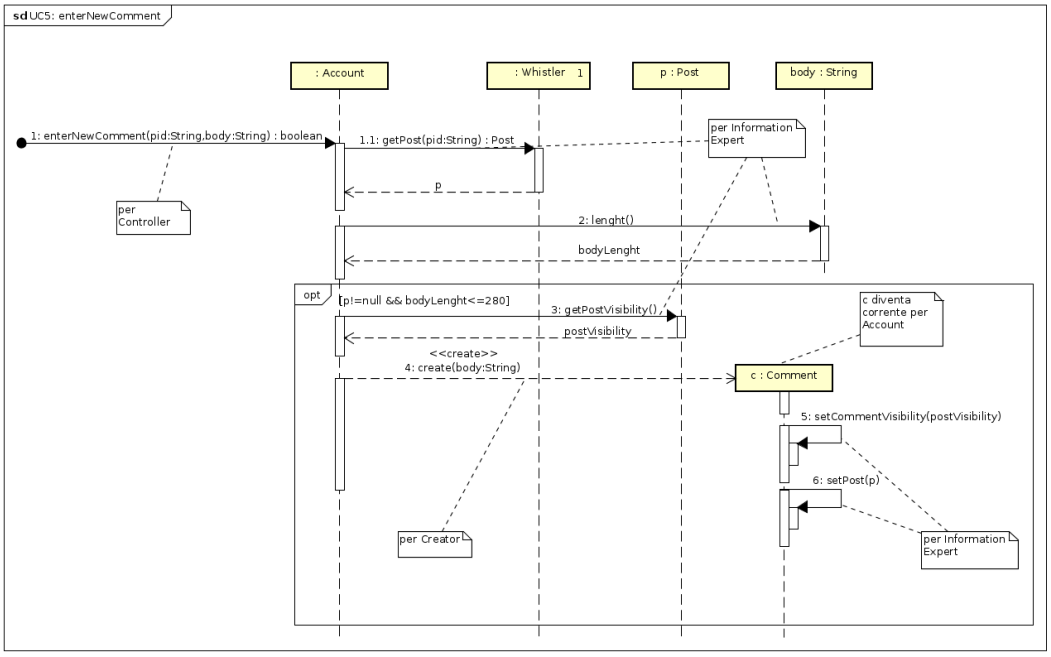
Diagrammi di Sequenza

- Ricerca di un post su Whistler

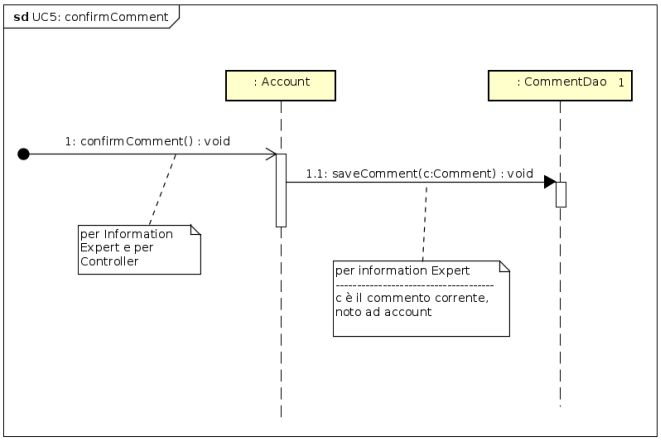




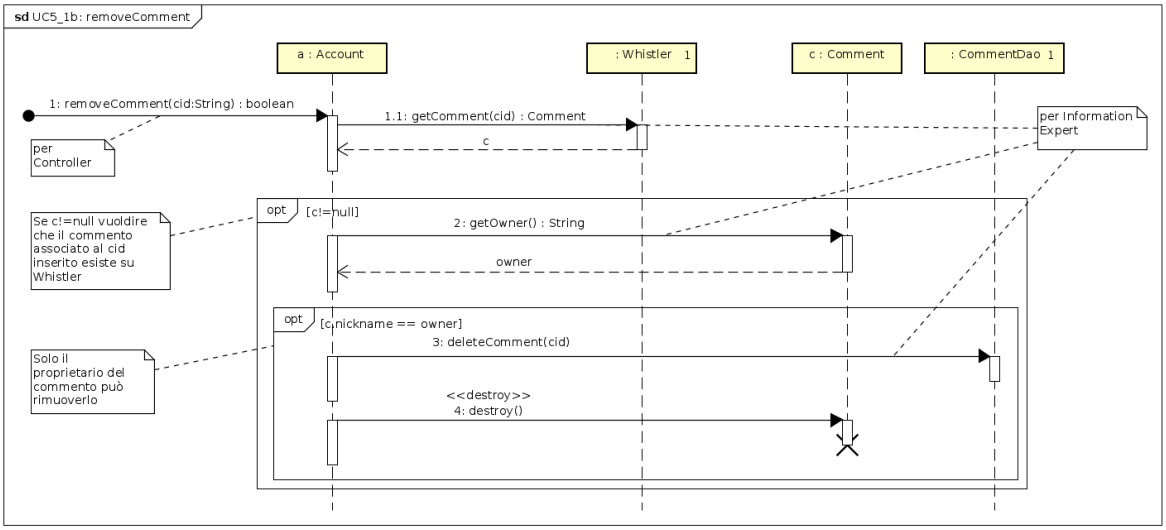
- Inserimento di un Commento su un Post presente su Whistler



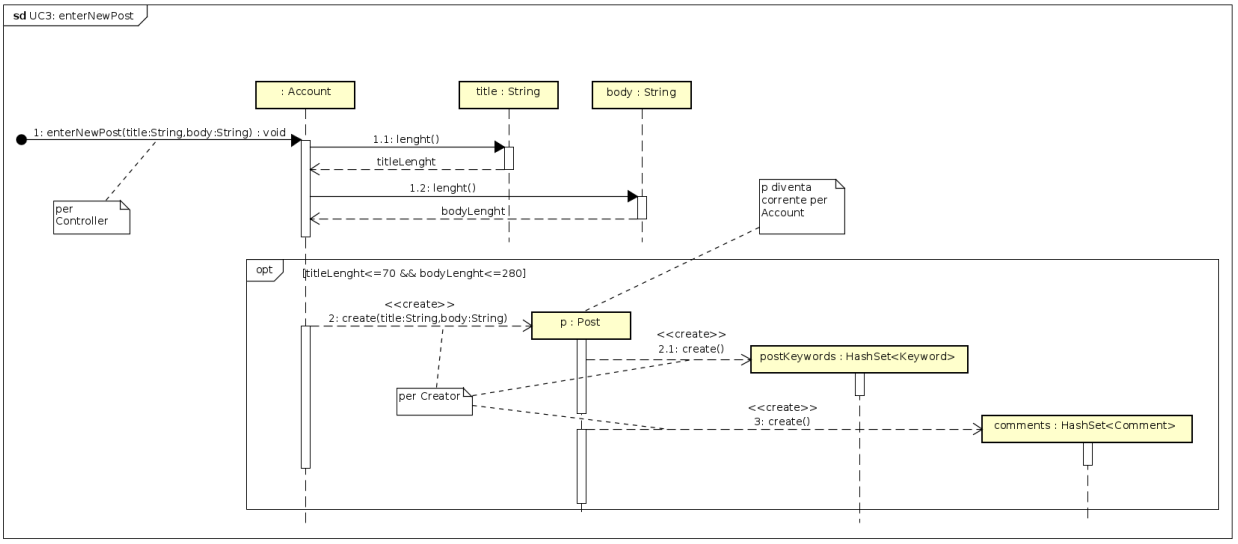
- Conferma creazione Commento



- Rimozione di un Commento



Si riporta di seguito la versione del Diagramma di Sequenza del caso d'uso **UC3 - enterNewPost** aggiornata all'iterazione corrente con il controllo sulla lunghezza del titolo e del corpo del post.



Diagrammi delle Classi

- DCD : Elaborazione 3

Si rimanda al file **Whistler_E3.asta** nella cartella 04_Elaborazione_3 per la consultazione del diagramma delle Classi relativo all'elaborazione 3.

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.

Classi di test:

Account Class
- 1) testEnterNewPost_TitleLenghtMoreThanSeventy **(CF)(VE)**
- 2) testEnterNewPost_TitleLenghtEqualSeventy **(CF)(VE)**
- 3) testEnterNewPost_TitleLenghtLessThanSeventy **(CF)(VE)**
- 4) testEnterNewPost_BodyLenghtMoreThanTwoHundredAndEighty **(CF)(VE)**
- 5) testEnterNewPost_BodyLenghtEqualTwoHundredAndEighty **(CF)(VE)**
- 6) testEnterNewPost_BodyLessThanTwoHundredAndEighty **(CF)(VE)**
- 7) testEnterNewComment_BodyLenghtMoreThanTwoHundredAndEighty **(CF)(VE)**
- 8) testEnterNewComment_BodyLenghtEqualTwoHundredAndEighty **(CF)(VE)**
- 9) testEnterNewComment_BodyLenghtLessThanTwoHundredAndEighty **(CF)(VE)**
- 10) testConfirmComment **(CF)**
- 11) testRemoveComment_owner **(CE)(CF)**
- 12) testRemoveComment_notOwner **(CE)(CF)**

E' stato inoltre aggiornato il test: *testRemovePost* in *testRemovePost_owner* ed è stato aggiunto *testRemovePost_notOwner*

- 11) testRemovePost_owner **(CE)(CF)**
- 12) testRemovePost_notOwner **(CE)(CF)**

Whistler Class
- 1) testSearchPosts **(CF)**
- 2) testGetKeyword_IsPresent() **(CE)(CF)**
- 3) testGetKeyword_NotPresent **(CE)(CF)**
- 4) testGetWhistlerPublicPosts **(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.

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.

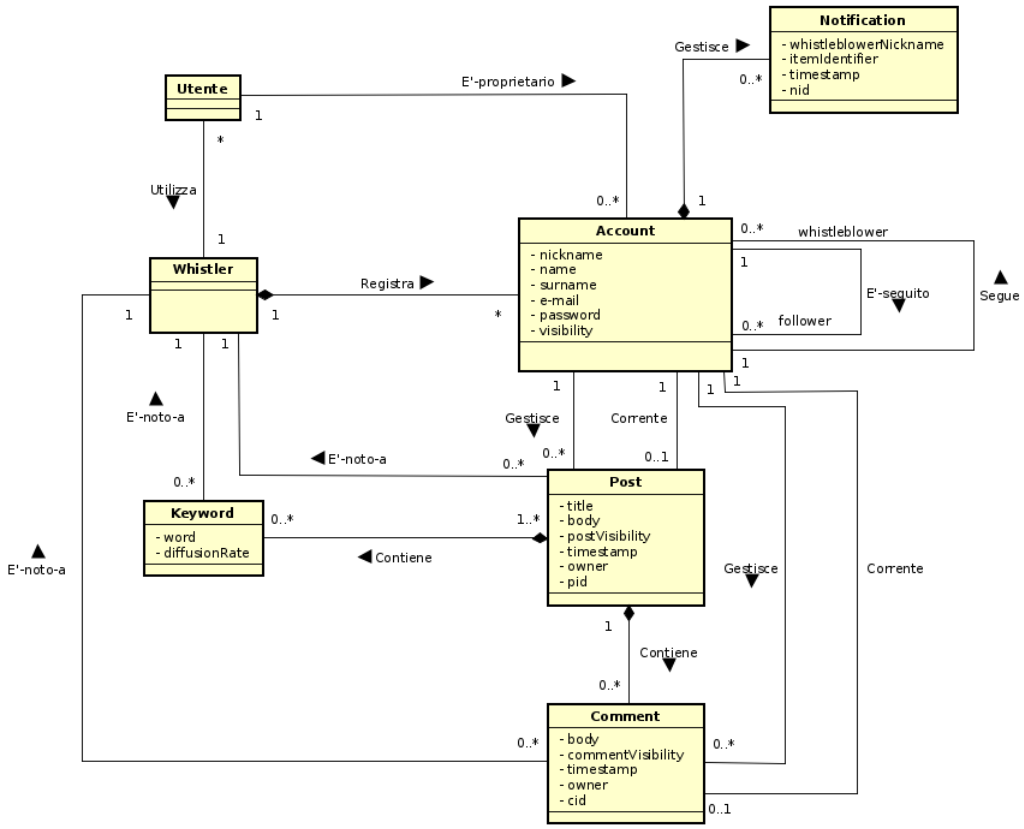
Modello di Dominio

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:

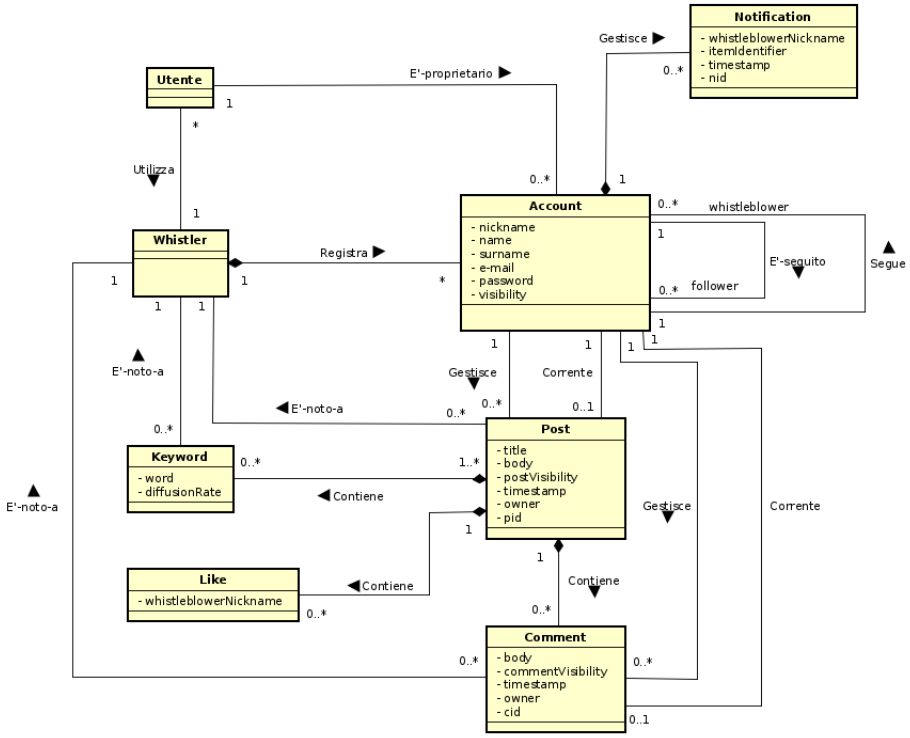
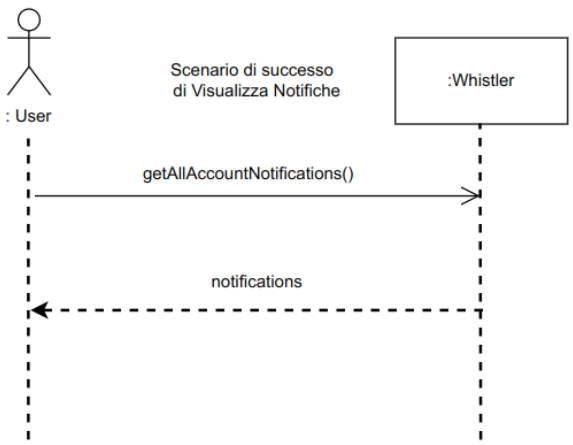


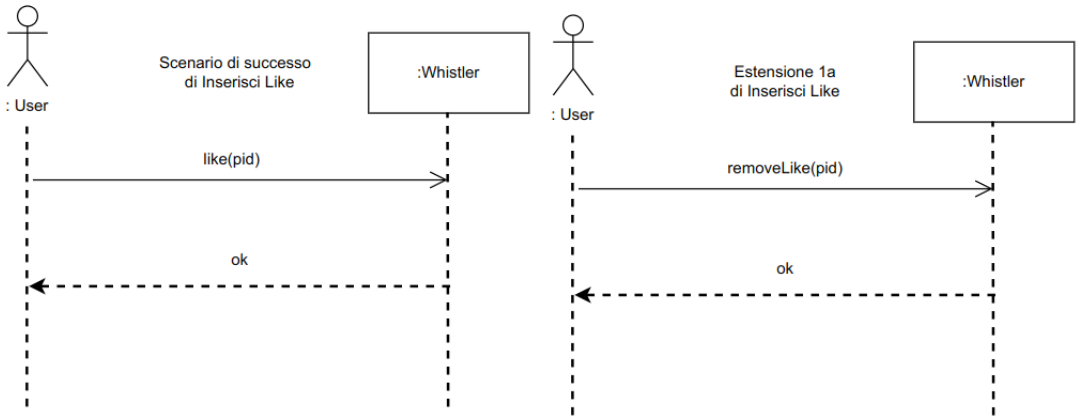
Diagramma di sequenza di sistema (SSD)

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:

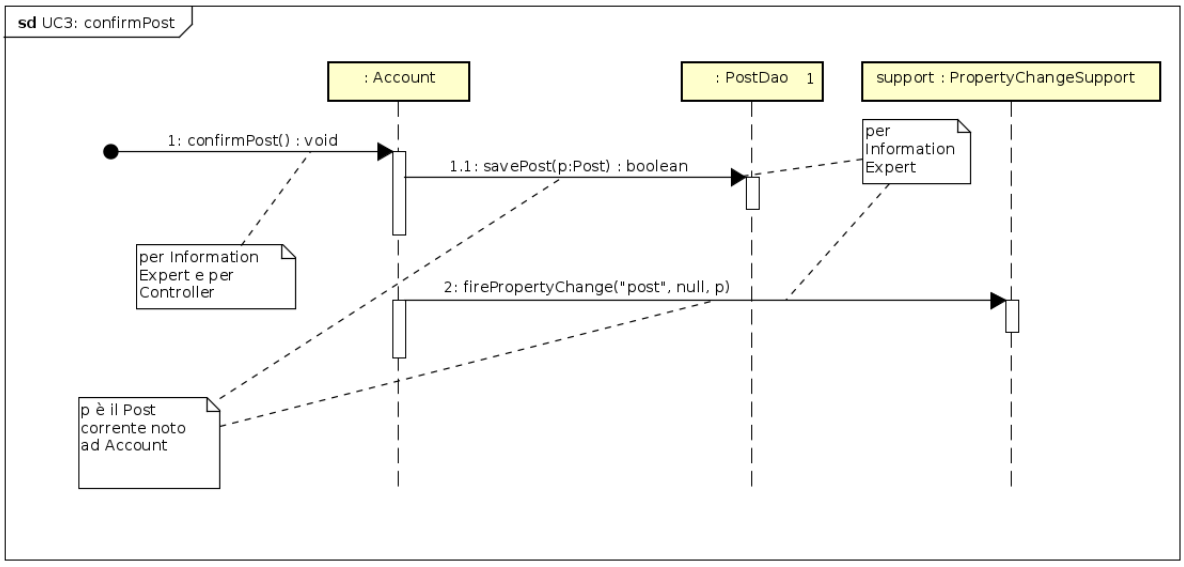
- presenti l'attributo ***support***
- implementi il metodo **addPropertyChangeListener**
  (necessario per l'aggiunta di Observers ad un Observable)
- implementi il metodo **removePropertyChangeListener**
  (necessario per la rimozione di Observers da un Observable)
- notifichi gli *Observers* richiamando 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:

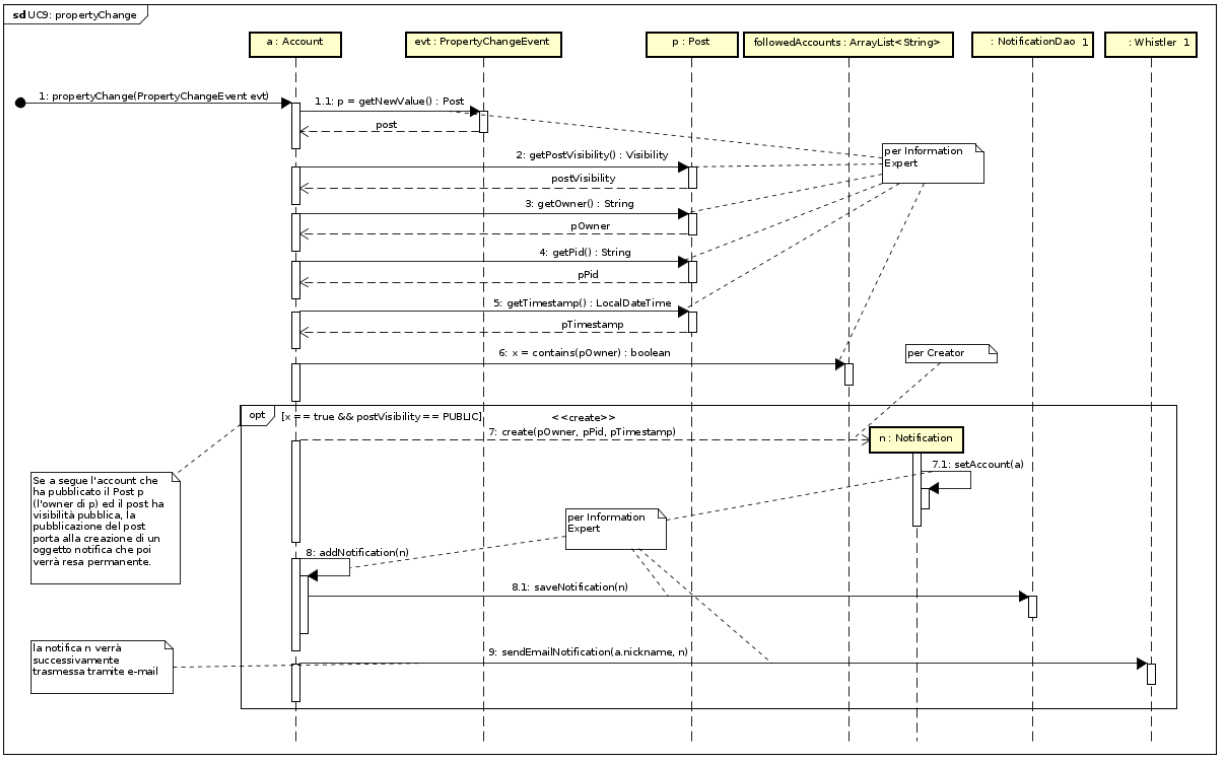
- implementi l'interfaccia ***PropertyChangeListener***
- implementi il metodo ***propertyChange***
  (che viene richiamato ogniqualvolta l'Observer riceve una notifica dall'Observable a cui è registrato)

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

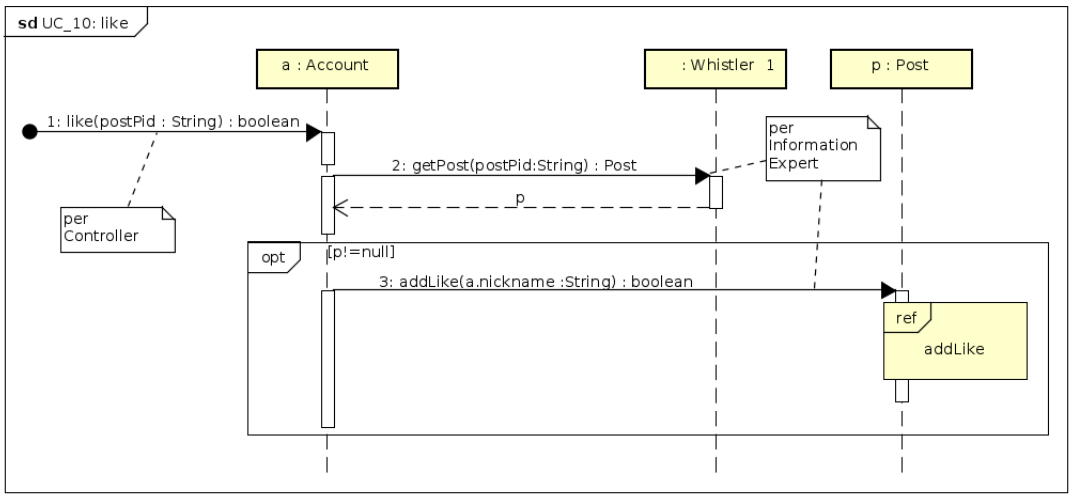


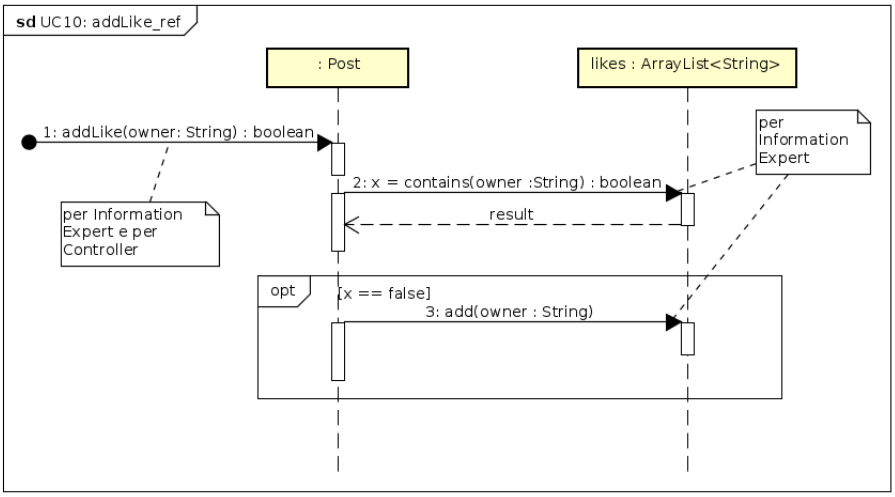
- Metodo chiamato lato follower alla pubblicazione di un nuovo Post



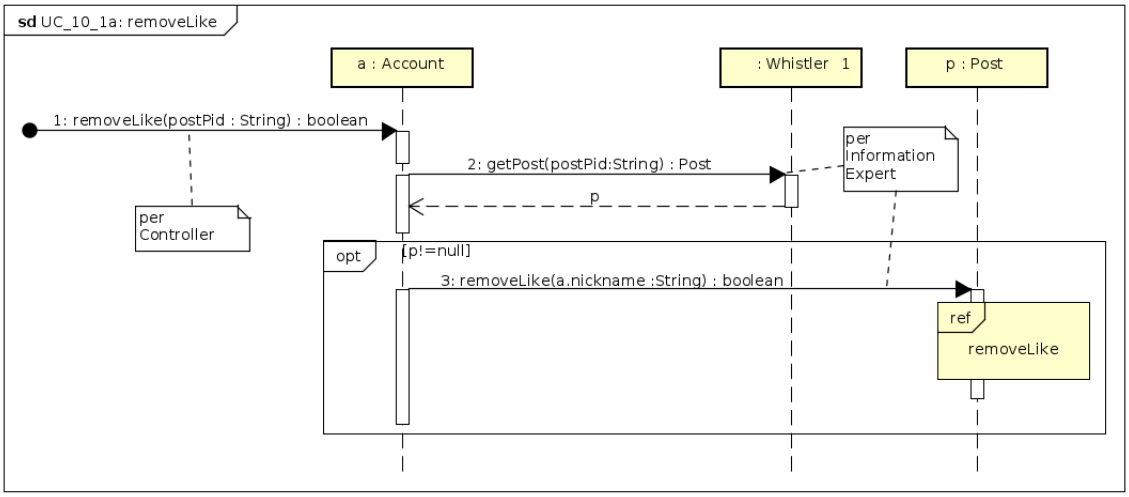
Si noti come la Notifica una volta resa permanente viene trasmessa anche per e-mail sfruttando l'indirizzo e-mail associato all'Account.

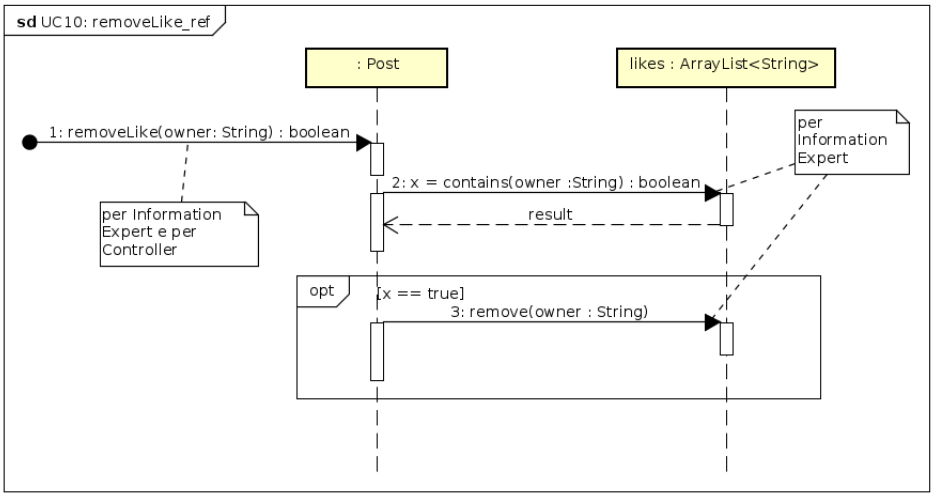
- Aggiunta di un Like





- 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.