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.

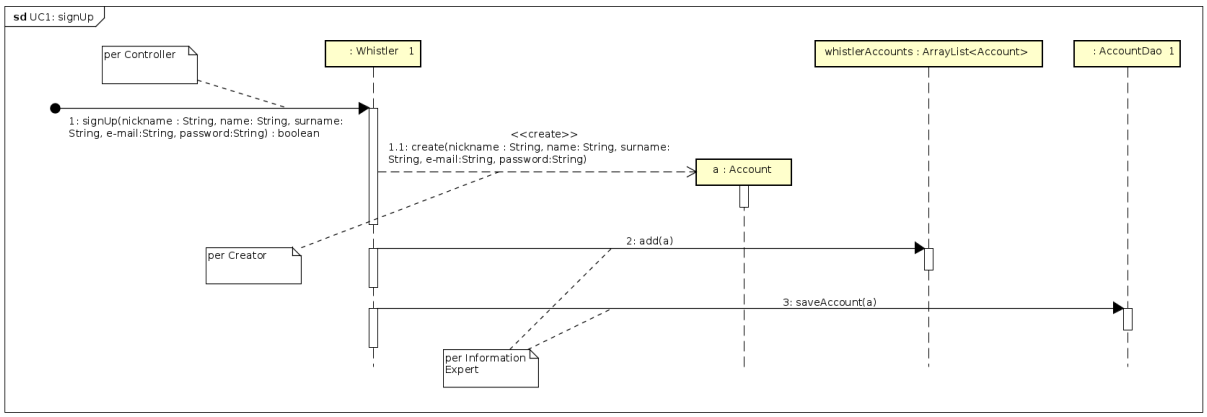
Per questa iterazione è stata realizzata la classe **AccountDao.java** ed il file **Account.hbm.xml** necessario per il mapping object/relational messo a disposizione da Hibernate, il quale è un'implementazione standard della **specifica JPA - Java Persistence API**. Il focus di JPA è quello di occuparsi 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.

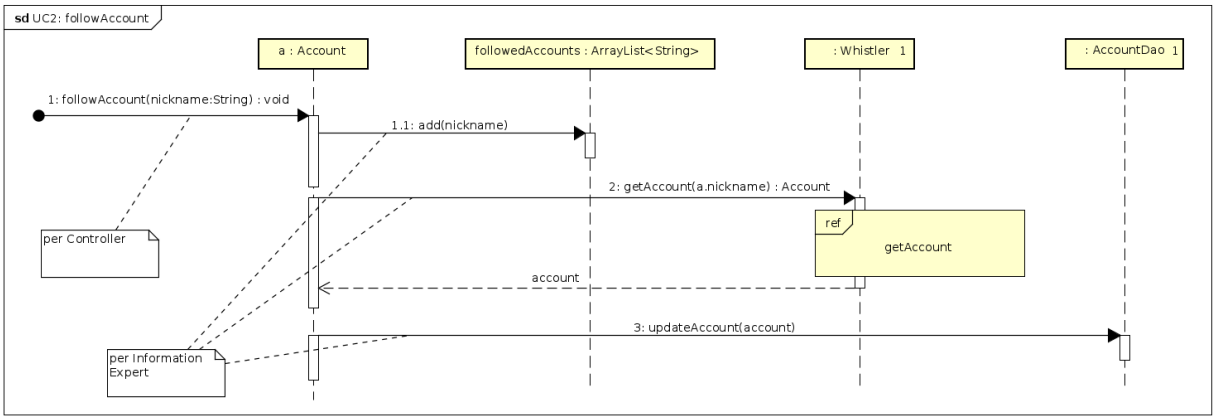
Di seguito si riportano i Diagrammi di Interazione (nello specifico **Diagrammi di Sequenza**) - modellazione dinamica - ed il **Diagramma delle classi** - modellazione statica - rivisti in seguito alle modifiche apportate.

Diagrammi di Sequenza

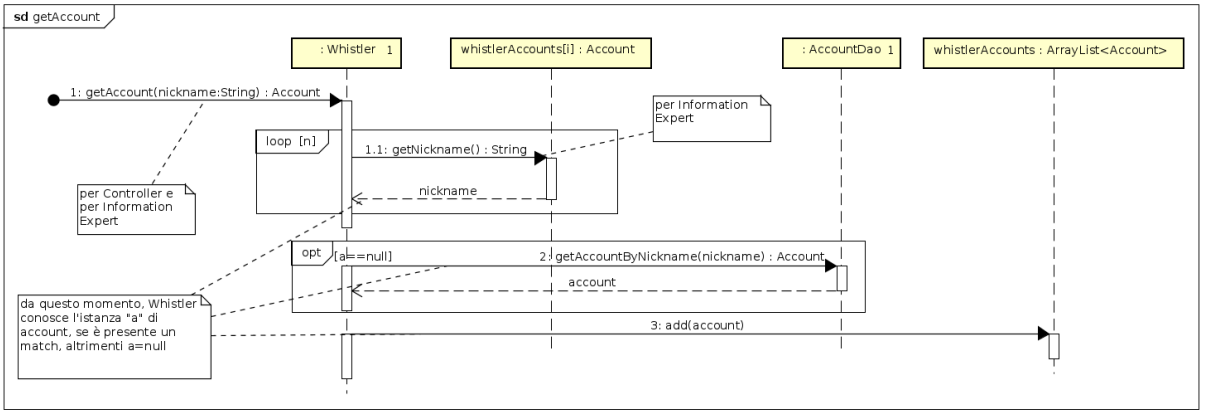
- Creazione di un account su Whistler



- Seguire un Account:



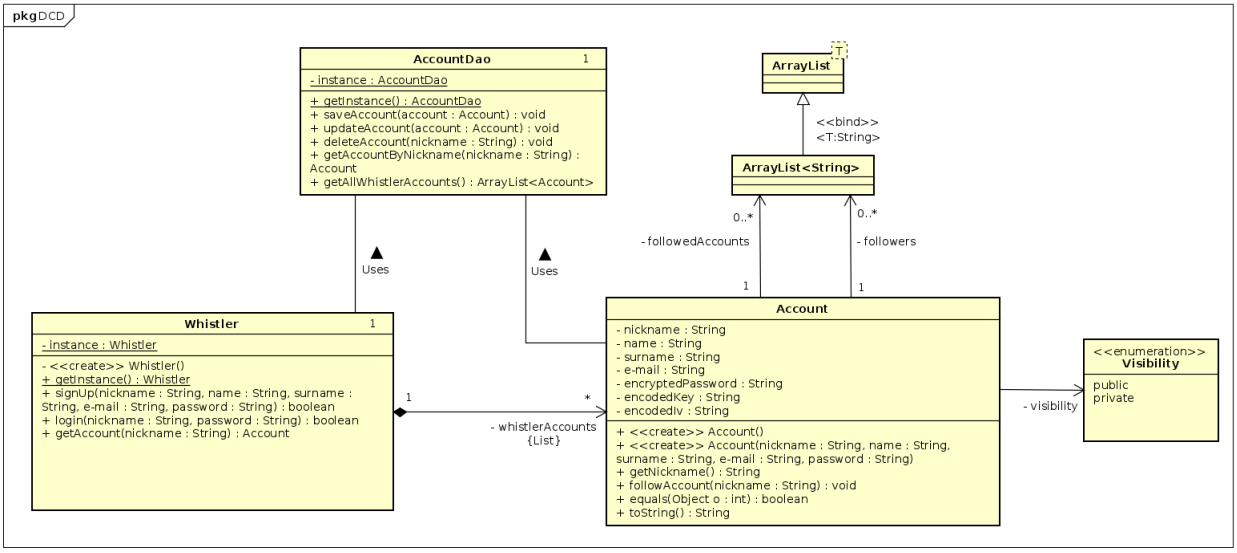
- getAccount:



[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.