document ID : deliverable_3

UNIVERSITY OF TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Progetto:

REVHUB

Titolo del documento:

Diagramma delle classi

Diagramma delle classi **1**
Scopo del documento **2**
1. Diagramma delle classi **3**
1.1. Utenti 3
1.2. Pubblicazione di contenuti 5
1.5. Diagramma delle classi complessivo 5
2. Codice in Object Constraint Language **7**
2.1 Creazione di una recensione 7
3. Diagramma delle classi con codice OCL **8**

Scopo del documento

Il presente documento riporta la definizione dell'architettura del progetto REVHUB usando diagrammi delle classi in Unified Modeling Language (UML) e codice in Object Constraint Language (OCL). Nel precedente documento è stato presentato il diagramma degli use case, il diagramma di contesto e quello dei componenti. Ora, tenendo conto di questa progettazione, viene definita l'architettura del sistema dettagliando da un lato le classi che dovranno essere implementate a livello di codice e dall'altro la logica che regola il comportamento del software. Le classi vengono rappresentate tramite un diagramma delle classi in linguaggio UML. La logica viene descritta in OCL perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

1. Diagramma delle classi

Nel presente capitolo vengono presentate le classi previste nell'ambito del progetto REVHUB. Ogni componente presente nel diagramma dei componenti diventa una o più classi. Tutte le classi individuate sono caratterizzate da un nome, una lista di attributi che identificano i dati gestiti dalla classe e una lista di metodi che definiscono le operazioni previste all'interno della classe. Ogni classe può essere anche associata ad altre classi e, tramite questa associazione, è possibile fornire informazioni su come le classi si relazionano tra loro.

Riportiamo di seguito le classi individuate a partire dai diagrammi di contesto e dei componenti. In questo processo si è proceduto anche nel massimizzare la coesione e minimizzare l'accoppiamento tra classi.

1.1. Utenti

Analizzando il diagramma di contesto realizzato per il progetto REVHUB si nota la presenza di **tre attori: "utente non autenticato", "amministratore"** e **"utente autenticato"**.

Lo "utente autenticato" è colei/colui che utilizza l'applicazione per poter creare recensioni e il "amministratore" è chi gestisce le segnalazioni alzate dagli utenti autenticati; mentre gli "utenti non autenticati" possono visualizzare le recensioni e fare login. Entrambi questi attori hanno specifiche funzioni e attributi ma hanno anche molto in comune. Sono state quindi individuate tre classi `UtenteAutenticato`, `UtenteNonAutenticato` `Amministratore` e con funzioni e attributi specifici e una classe `Utente` con funzioni e attributi in comune collegate tramite una generalizzazione. Nella classe `UtenteNonAutenticato` ci sono in più i metodi di login e di creazione di un profilo sono state aggiunte. Successivamente la classe `UtenteAutenticato` eredita tutti i campi e metodi `Utente` da con in più attributi come Nome, Cognome, Username che viene scelto dall'utente, Email, Id e Password. Infine c'è `Amministratore` che è una specializzazione dell'utente Autenticato in grado di gestire, secondo il regolamento e i Termini e Condizioni, le segnalazioni.

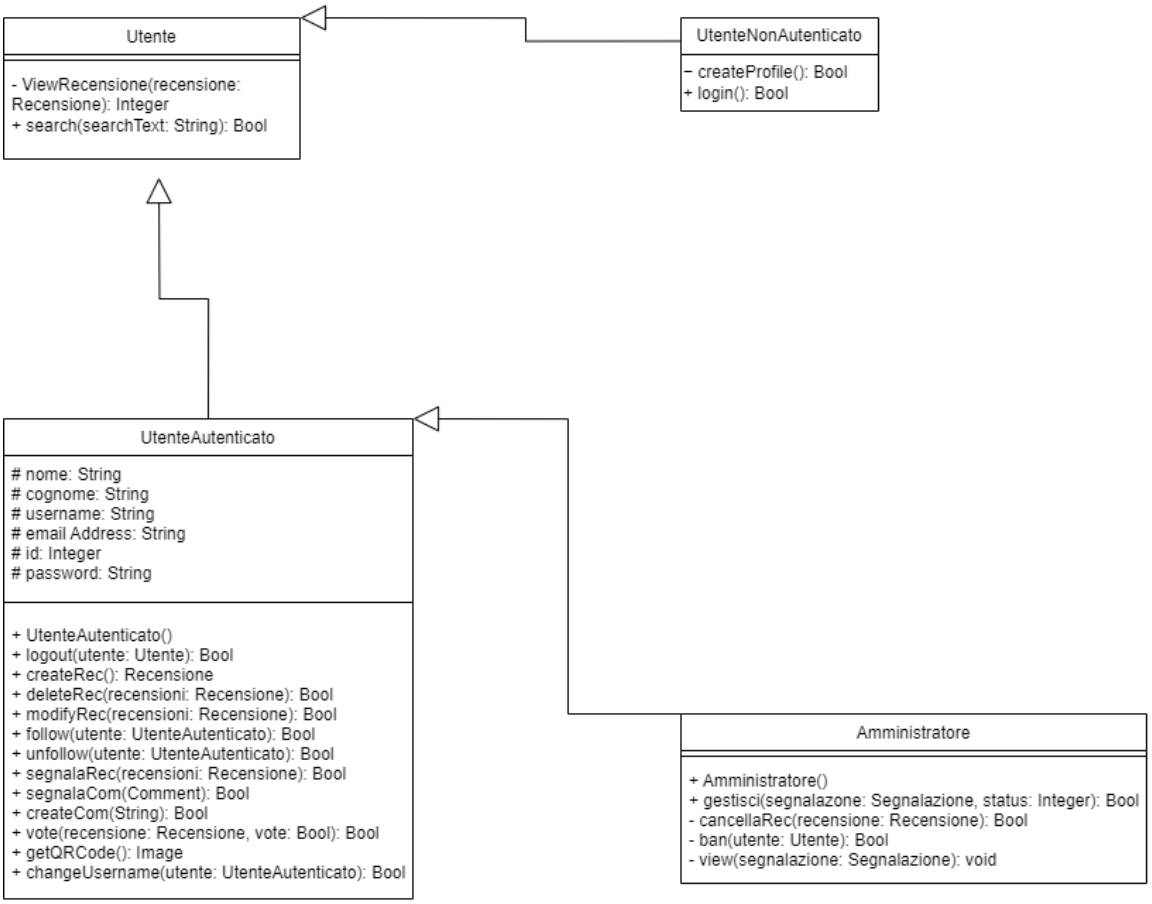

Figura 1.0 Utenti

1.2. Pubblicazione di contenuti

Il componente **Pubblicazione di contenuti** rappresenta la possibilità per un Utente Autenticato di creare Recensioni e Commenti, i quali sono entrambi specializzazioni della classe Post. In questo modo viene descritta anche la struttura e gli attributi di una recensione.
Inoltre è possibile notare come entrambe le classi Utente Autenticato e Recensione implementino l'interfaccia Searchable, che permette loro di essere cercate tramite la barra di ricerca.

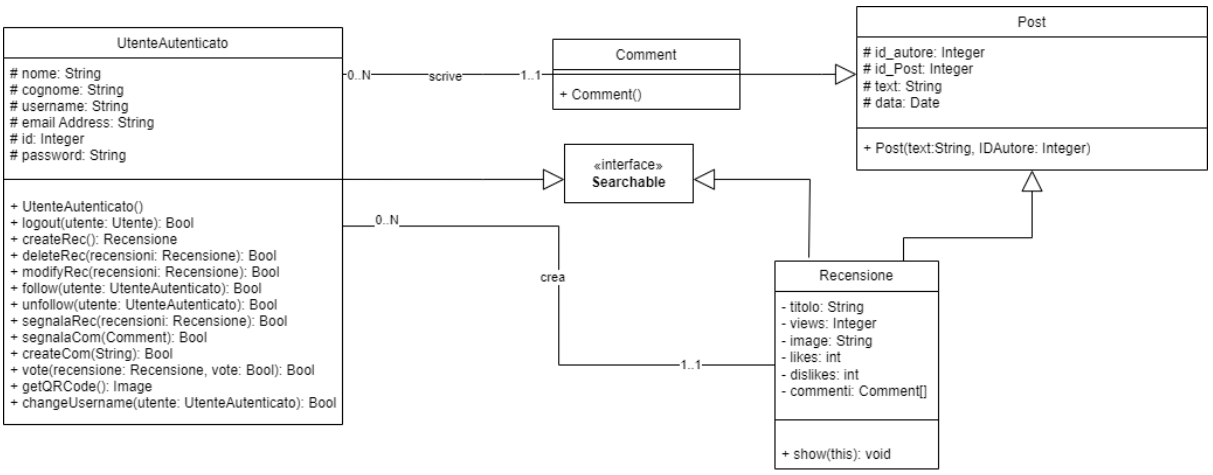

Figura 2.0 Pubblicazione Contenuti

1.3. Ricerca

Un utente, anche non registrato, quando utilizza il programma, ha a disposizione una barra di ricerca in cui può inserire una stringa di testo per poter cercare:

- **utenti** registrati, tramite il proprio username univoco

- delle **recensioni**, tramite il titolo

La barra di ricerca va a chiamare il metodo ~find(String str)~ che elabora la stringa inserita dall'utente, e restituisce tutte le corrispondenze che riesce a trovare sul database.

Se la stringa di ricerca fornita dall'utente inizia con il carattere `@` e non è composta da molteplici parole, il metodo `find()` restituisce una lista di utenti i cui username sono simili a quello fornito, ovvero con una distanza di Levenshtein minore o uguale a 10.

Altrimenti, nel caso in cui l'utente non stia cercando un utente, il metodo restituisce una lista di recensioni che contengono al loro interno la stringa di testo cercata. La ricerca su database verrà fatta secondo l'algoritmo tf-idf

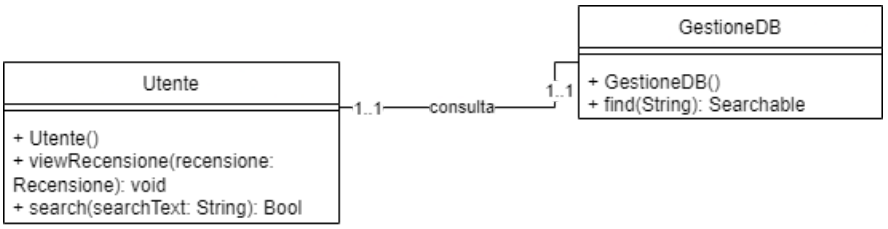

Figura 3.0 Ricerca

1.4. Segnalazioni

Un utente autenticato ha la possibilità di segnalare una specifica recensione, fornendo una motivazione.

Un utente amministratore ha la possibilità di gestire una segnalazione creata da un utente autenticato. La segnalazione è gestita tramite il metodo `gestisci(segnalazione: Segnalazione, status: Integer)`.

Sulla base della scelta dell'amministratore, il metodo potrebbe chiamare uno dei seguenti metodi privati:

- `ban(utente: Utente)`, metodo che banna l'utente che ha creato la recensione

- `cancellaRec(recensione: Recensione)`, metodo che cancella la recensione dal database, con tutti i commenti annessi

I valori di `utente` e `recensione` sono ricavabili dalla segnalazione fornita al metodo `gestisci()`.

Questi metodi andranno a modificare lo stato in cui si trova la recensione, che potrebbe essere uno dei seguenti:

- *0*: recensione non gestita

- *1*: recensione gestita, l'amministratore ha riscontrato un falso positivo e

  non ha effettuato nessuna operazione

- *2*: recensione gestita, recensione cancellata

- *3*: recensione gestita, l'amministratore ha bannato l'utente che ha scritto la recensione

- *4*: recensione gestita, l'amministratore ha bannato l'utente che ha scritto la segnalazione

Il caso *4* è utilizzato quando uno stesso utente autenticato crea molteplici segnalazioni in modo malevolo.

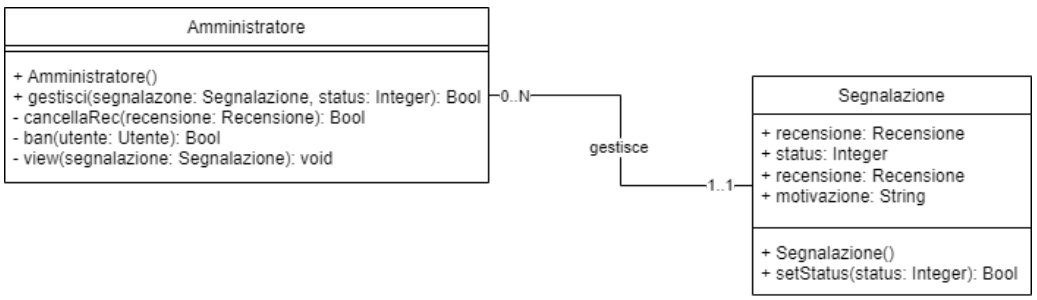

Figura 4.0 Segnalazioni

1.5. Diagramma delle classi complessivo

Riportiamo di seguito il diagramma delle classi con tutte le classi fino ad ora presentate.

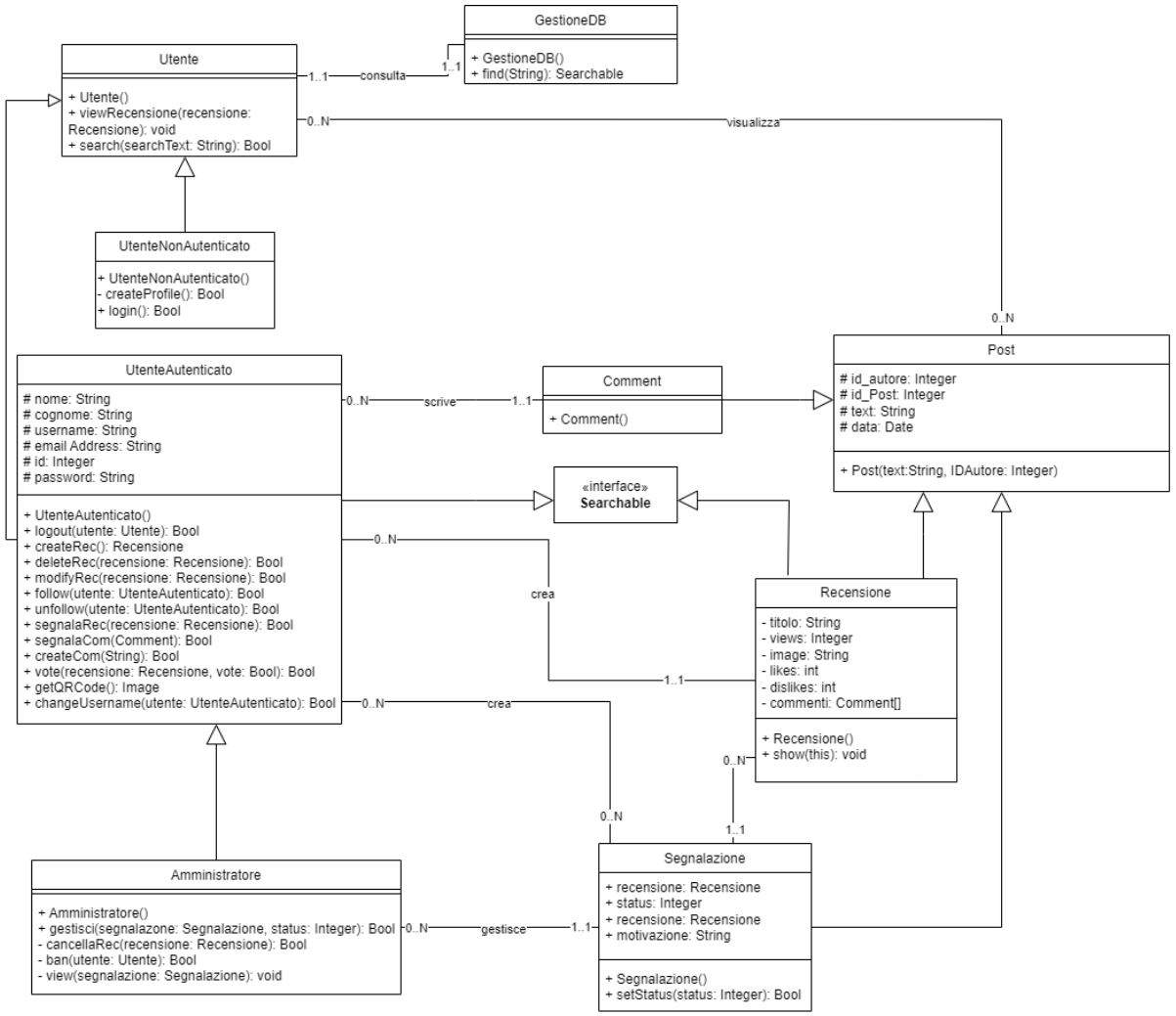

Figura 5.0 Diagramma delle Classi

2. Codice in Object Constraint Language

In questo capitolo è descritta in modo formale la logica prevista nell'ambito di alcune operazioni di alcune classi. Tale logica viene descritta in Object Constraint Language (OCL), poiché tali concetti non sono esprimibili in modo formale nel contesto di UML.

2.1 Creazione di una recensione

Un utente autenticato per creare una recensione deve assicurarsi di compilare tutti i suoi campi obbligatori, come espresso dai seguenti vincoli:

- context UtenteAutenticato::CreateRec()
  pre: titolo != “”
  pre: text != “”

2.2 Creazione di un profilo

- context UtenteNonAutenticato::createProfile()
  post: nome != “”
  post: cognome != “”
  post: username != “”
  post: email Address != “”
  post: id != “”
  post: password != “”

2.3 Visualizzazioni

- context Utente::ViewRecensione()
  post: Recensione.views = Recensione.views + 1

2.4 Gestione segnalazioni

- context Amministratore::gestisci()
  post: segnalazione.setGestito()

3.Diagramma delle classi con codice OCL

Riportiamo infine il diagramma delle classi con tutte le classi fino ad ora presentate ed il codice OCL individuato.

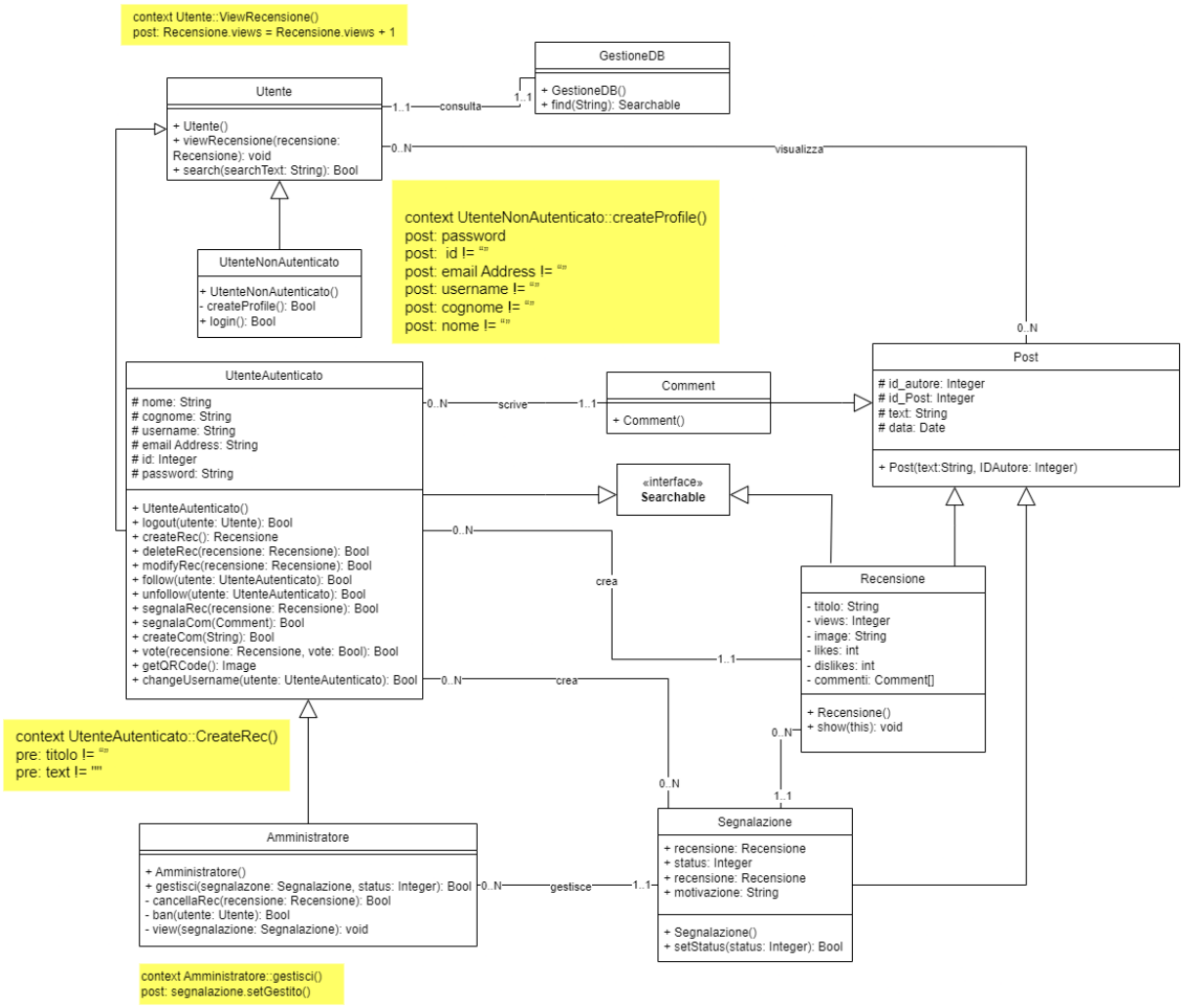

Figura 6.0 Diagramma delle Classi con OCL