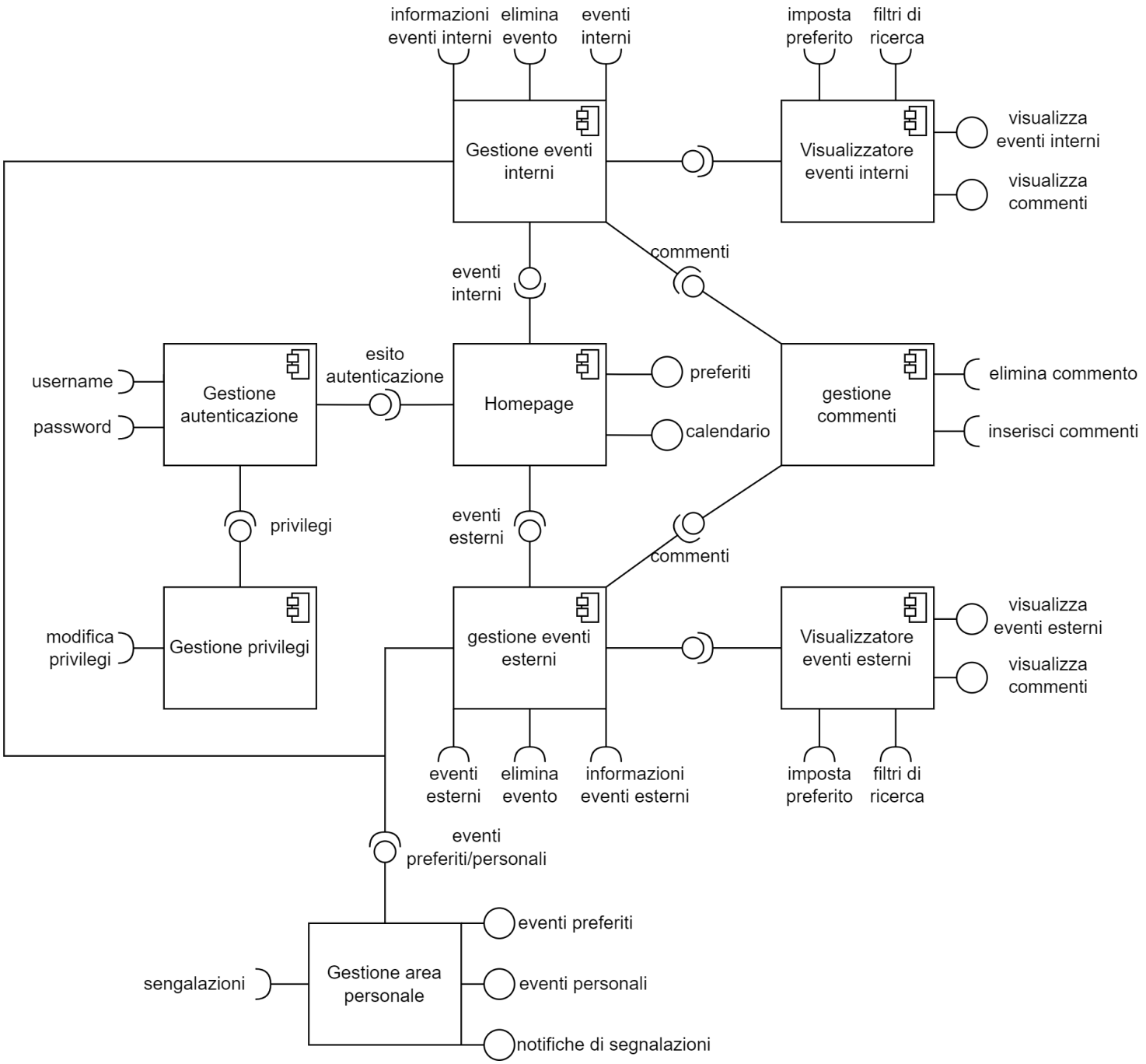
UNiVENTS

Component,
Class Diagrams,
OCL.

Doc. Name	Progetto_IS_UNiVENTS_3	*Doc. Number*	C1
Description	Documento di progetto: Component Diagram, Class Diagram		
Gruppo	Aloisi Deborah - 209557 Pasetto Davide - 209485 Tomasi Elia - 205577		

INDICE

COMPONENT DIAGRAM



ANALISI DEI COMPONENTI

1: Gestione autenticazione

La nostra applicazione sfrutta la pagina di login dell'università di Trento per effettuare l'autenticazione degli utenti (come specificato nel **RF1**) . E' necessario quindi un componente "Gestione autenticazione" che si interfacci con il sistema dell'ateneo per effettuare il login e gestire le richieste di accesso all'applicazione, mantenendo poi attiva la sessione per uno studente autenticato.

Coesione: livello 7 - *funzionale*

Ogni elemento del componente lavora allo scopo di gestire gli accessi e il sistema di autenticazione.

2: Gestione privilegi

Questo componente gestisce i privilegi degli utenti, che definiscono le azioni che gli utenti possono eseguire all'interno dell'applicazione. Tali privilegi vengono forniti al componente "Gestione autenticazione" una volta effettuato con successo il login. L'amministratore di sistema può decidere di modificare i privilegi degli utenti in ogni momento (**RF11**). Questi privilegi identificheranno ogni utente come "Studente", "Moderatore" o "Event manager", eventualmente sbloccando nuove funzioni.

Coesione: livello 7 - *funzionale*

Lo scopo del componente è quello di gestire i privilegi e le autorizzazioni degli utenti.

3: Homepage

La "Homepage" è un componente che si occupa di mostrare all'utente alcuni eventi particolari (come gli eventi di oggi e i preferiti) e il calendario. Inoltre dalla home page è possibile navigare verso altre pagine all'interno dell'applicazione (come la pagina di visualizzazione degli eventi).

Coesione: livello 2 - *logico*

Questo componente non ha un unico scopo ben definito, se non quello di mostrare all'utente alcune categorie di eventi e di fare da crocevia verso altre sezioni dell'applicazione.

4/5: Gestione eventi interni/esterni

N.B.: Nonostante i due componenti “Gestione eventi interni” e “Gestione eventi esterni” gestiscano due categorie diverse di eventi, essi si comportano di fatto allo stesso modo, e verranno pertanto trattati contemporaneamente.

I due componenti “Gestione Eventi” si occupano di gestire gli eventi memorizzati nel database, permettendo la creazione di nuovi eventi e l’eventuale eliminazione degli stessi, interfacciandosi con i database.

Questi due componenti soddisfano i requisiti funzionali **RF2** (nel caso degli eventi interni all’ateneo), **RF3** (per gli eventi esterni), **RF4** e **RF12** (creazione degli eventi) e **RF9** (eliminazione degli eventi) .

N.B.: per quanto riguarda la creazione di un evento, gli attributi “Informazioni eventi” sono le informazioni caricate dagli utenti durante la creazione di un evento, mentre gli “eventi interni/esterni” sono gli eventi già esistenti recuperati dal database

Coesione: livello 7 - *funzionale*

Tutti gli elementi dei due componenti collaborano allo scopo di gestire la creazione e l’eliminazione degli eventi.

6/7: Visualizzatore eventi interni/esterni

N.B.: Come per i due componenti di Gestione eventi, anche questi due componenti di visualizzazione verranno trattati allo stesso modo.

Questi due componenti, “Visualizzatore eventi interni” e “Visualizzatore eventi esterni”, forniscono all’utente delle interfacce grafiche per la visualizzazione degli eventi e delle informazioni ad essi correlate (**RF5**), nonché dei commenti pubblicati dagli utenti stessi. Questi componenti ricevono le informazioni di eventi e commenti dal componente di “Gestione eventi” rispettivo. Saranno anche disponibili alcune opzioni per impostare dei filtri per effettuare una ricerca specifica di uno o più eventi (**RF10**)

Da queste interfacce grafiche sarà inoltre possibile impostare un evento come preferito (**RF6**), segnalare un evento o un commento (**RF8**)

Coesione: livello 7 - *Funzionale*

Scopo di questo componente è far visualizzare all’utente gli eventi e le rispettive informazioni

8: Gestione commenti

Questo componente si occupa di permettere la creazione dei commenti attraverso una pagina specifica a cui si può accedere attraverso un pulsante nella schermata di visualizzazione di un evento (**RF7**). Se l'utente dispone dei privilegi di moderatore, sarà possibile per lui anche l'opzione di eliminare uno o più commenti (**RF9**).

Coesione: livello 7 - *Funzionale*

Il componente lavora per la gestione dei commenti e la loro creazione/eliminazione.

9: Gestione area personale

Questo componente gestisce l'area personale dell'utente, una sezione dell'applicazione dove sarà possibile visualizzare gli eventi preferiti, gli eventi creati dall'utente stesso e, nel caso si possedessero i privilegi di moderatore, le notifiche delle segnalazioni effettuate dagli utenti. Da questa pagina sarà inoltre possibile eseguire il logout per terminare la sessione e disconnettersi dall'applicazione.

Coesione: livello 2 - *logica*

Il componente è composto da diverse sezioni correlate ma staccate tra di loro, di cui una sola verrà selezionata dall'utente.

Componenti/ Livelli di coesione	1 Casuale	2 Logica	3 Temporale	4 Procedurale	5 Comunicazione	6 Informazionale	7 Funzionale
Gestione Autenticazione							x
Gestione Privilegi							x
Homepage		x					
Gestione eventi							x
Visualizzatore eventi							x
Gestione commenti							x
Gestione area personale		x					

LIVELLO DI ACCOPPIAMENTO TRA COMPONENTI

COMPONENTI	1	2	3	4	5	6	7	8	9
1		data	data	x	x	x	x	x	x
2			x	x	x	x	x	x	x
3				data	data	x	x	x	x
4					x	data	x	data	data
5						x	data	data	data
6							x	x	x
7								x	x
8									x
9									

ANALISI DEGLI ACCOPPIAMENTI

1: Gestione autenticazione - Gestione permessi, livello data

Lo scambio che avviene tra questi due componenti è quello delle informazioni sui permessi associati all'utente che sta facendo il login e sulla sessione.

2: Gestione autenticazione - Homepage, livello data

Il componente "gestione autenticazione" passa alla Homepage informazioni sull'esito del login e sui permessi dell'utente che ha aperto la sessione.

3: Homepage - Gestione eventi interni, livello data

Il modulo "Gestione eventi interni" invia al componente "Homepage" informazioni sugli eventi che la homepage deve mostrare all'utente.

4: Homepage - Gestione eventi esterni, livello data

Come per l'analisi dell'accoppiamento n° 3, anche qua vengono scambiati informazioni sugli eventi.

5: Gestione eventi interni - Visualizzatore eventi interni, livello data

Il componente "Gestione eventi interni" invia al "Visualizzatore eventi interni" informazioni su tutti gli eventi da visualizzare e sui commenti pubblicati sotto tali eventi.

6: Gestione eventi interni - Gestione commenti, livello data

Il modulo "Gestione commenti", dopo aver permesso la creazione di un commento, ne invia le informazioni alla "Gestione eventi interni".

7: Gestione eventi interni - Gestione area personale, livello data

Il modulo "Gestione eventi interni" invia le informazioni sugli eventi preferiti e sugli eventi di oggi al modulo "Gestione area personale", perché quest'ultimo possa mostrarli all'utente.

8: Gestione eventi esterni - Visualizzatore eventi esterni, livello data

Come per l'accoppiamento n° 5, solo che ad essere trattati sono gli eventi esterni invece di quelli interni.

9: Gestione eventi esterni - Gestione commenti, livello data

Come l'accoppiamento n° 6.

10: Gestione eventi esterni - Gestione area personale, livello data

Come l'accoppiamento n° 7.

N.B.: Ogni volta che un utente naviga all'interno dell'applicazione da una pagina ad un'altra (incluso il passaggio dalla pagina di login alla homepage), verrà effettuato un controllo sui privilegi dell'utente per decidere se mostrare o meno alcune azioni riservate.

CLASS DIAGRAM

1 - Utenti

Nel nostro progetto vi sono principalmente 3 attori; gli *studenti*, i *moderatori* e gli *event manager*.

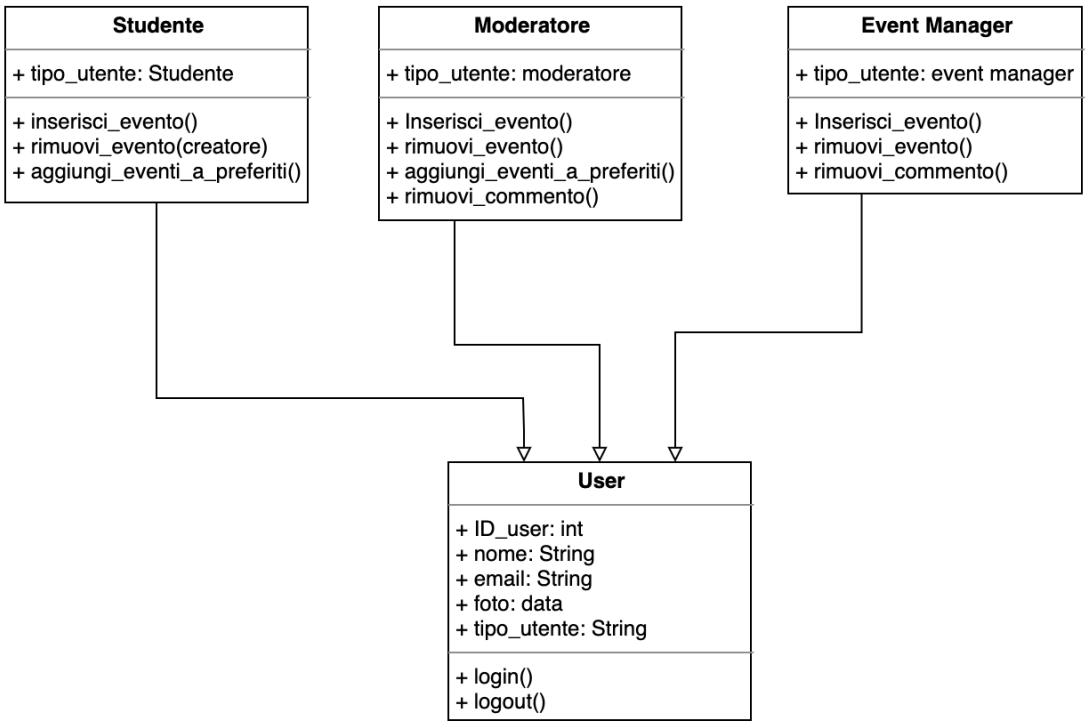
Tra questi tre attori sono stati individuati degli attributi e delle funzioni comuni collegati tramite una generalizzazione (classe *utente*).

Nella classe *moderatore* sono state aggiunte le operazioni di moderazione degli eventi, mentre nella classe di *event manager* sono stati aggiunti la gestione degli eventi interni.

La classe *studente* ha la possibilità di rimuovere un evento solo se l'utente che vuole eseguire l'operazione ne è il creatore. Invece la classe *moderatore* può rimuovere un evento a prescindere da chi ne sia il creatore perché ricade nel requisito funzionale RF9 .

La classe *event manager* invece ha la possibilità di inserire e rimuovere solo gli eventi classificati come interni (eventi a cura organizzativa dell'università).

Per gestire ciò viene assegnato ad ogni utente una variabile tipo_utente che li differenzia nei vari attori. La variabile può essere modificata soltanto dall'amministratore di sistema che assegna i ruoli agli utenti.



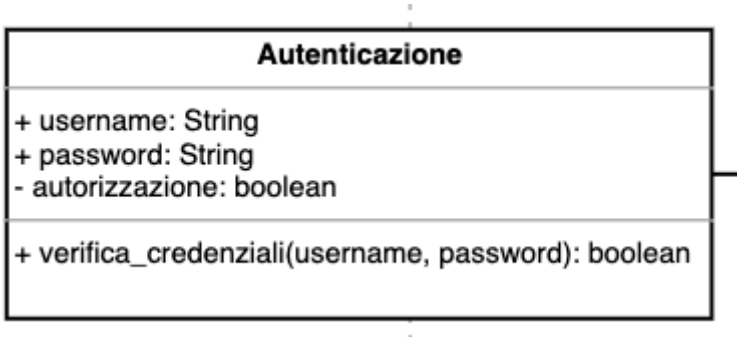
2 - Autenticazione

Gli utenti si autenticano con un sistema esterno di gestione delle credenziali universitarie. Individuiamo così la classe *autenticazione*.

L'applicazione non gestirà e non memorizzerà username e password inserite dall'utente ma con questa classe passerà solo i dati di autenticazione al sistema subordinatosterno, il sistema credenziali universitarie, che valuterà questi dati e risponderà specificando se le credenziali inserite sono valide o meno.

Dopo l'autenticazione, la componente di gestione privilegi, in base alla tipologia dell'utente, permette di prendere visione della propria 'home page'.

Di seguito il dettaglio di questa classe con i propri attributi e metodi.



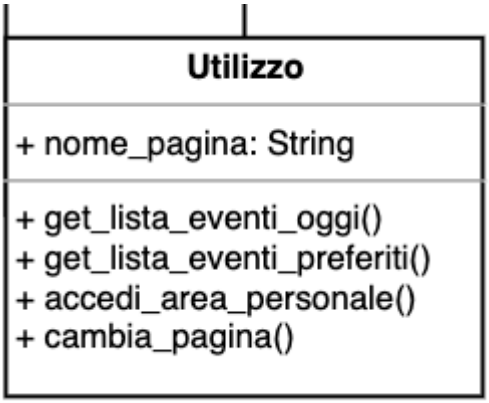
3 - Homepage

Nella componente homepage identifichiamo una classe; *Utilizzo*.

La classe user potrà compiere le seguenti azioni: spostarsi di pagina, visionare gli eventi di oggi, visionare i suoi eventi preferiti, entrare nel suo profilo personale.

La variabile ‘nome_pagina’ si riempie con il nome della pagina che l’utente attualmente visiona; una volta che l’utente richiama la funzione cambia pagina la variabile viene aggiornata con la nuova pagina. La variabile serve per gestire lo spostamento tra le pagine.

L’operazione ‘accedi_area_personale()’ permette all’utente di avere una preview delle funzionalità nell’area personale come descritto nel RF6.



4 - Gestione eventi

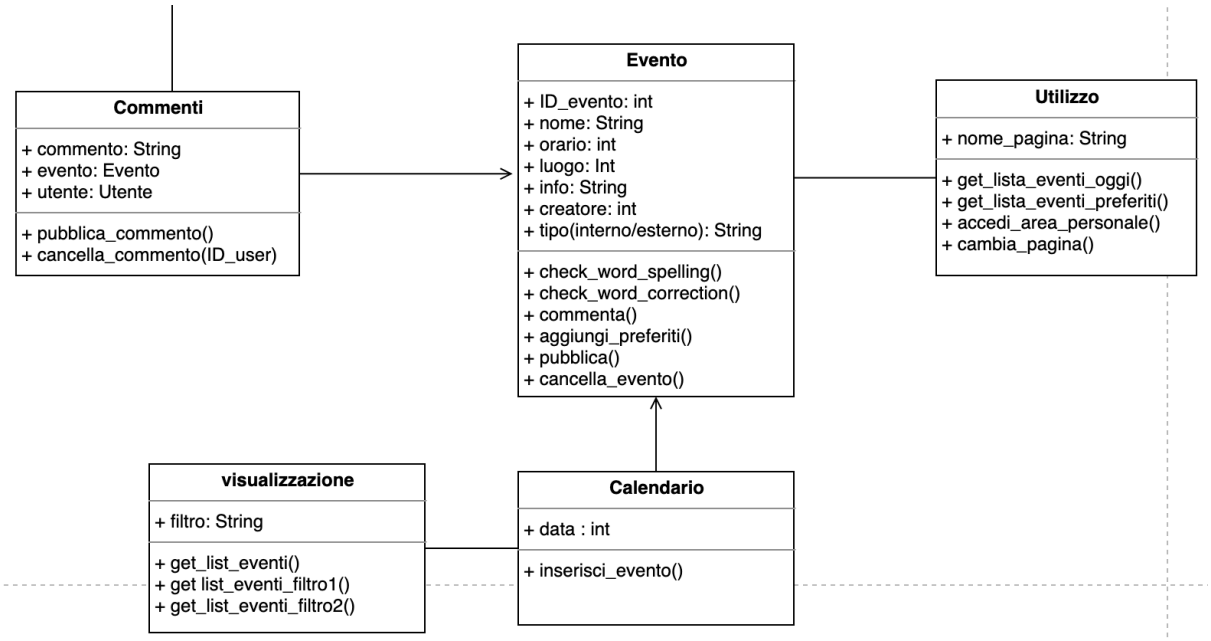
Nella componente di gestione eventi individuiamo la classe *evento,* la classe *calendario*, la classe *visualizzazione* e la classe *commenti* .

La classe *evento* si occupa di immagazzinare le informazioni relative ad un evento e li classifica come eventi interni o esterni in base al tipo dell'evento. Quando si utilizza la funzione 'commenta()' viene creato un nuovo oggetto di classe 'commento' le cui variabili di evento e di utente si riempiono con l'evento che la chiama e l'utente attualmente loggato.

La classe *calendario* si occupa di inserire gli eventi forniti in un ordine cronologico in base alla data di avvenimento. Quando un evento viene creato, la funzione 'pubblica()' viene chiamata richiama le funzioni di check e alla fine richiama la funzione 'inserisci_evento()' che inserisce l'evento nella data e orario forniti. Nella variabile 'data' invece viene salvata la data attuale.

La classe *Visualizzazione* filtra gli eventi in base al parametro di ricerca utilizzato, per default mostra gli eventi nell'ordine di inserimento nel database. In base alla variabile 'filtro' la classe chiama le diverse funzioni che ordinano gli eventi in base al filtro fornito.

La classe *commenti* associa una stringa (commento) a un evento e un utente. Il metodo 'pubblica_commento()' inserisce il commento nel database e l'operazione 'rimuovi_commento' invece rimuove il commento solo se chi esegue l'operazione ne è l'autore



5 - Amministratore

La classe amministratore è una classe relativa alla classe user ma possiede tutta la parte di gestione dei privilegi di altri utenti.

Possiede i privilegi di nominare o rimuovere qualsiasi tipo di utente, di rimuovere qualsiasi tipo di evento e commento.

La sua funzione principale è quella di assegnare i ruoli agli altri utenti e quindi di modificare la variabile 'tipo_utente'.

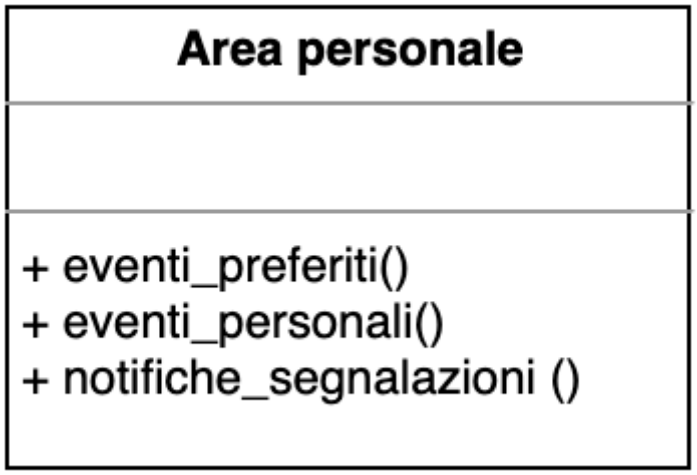
Può nominare anche altri amministratori nel caso vi fosse necessità.

6 - Area Personale

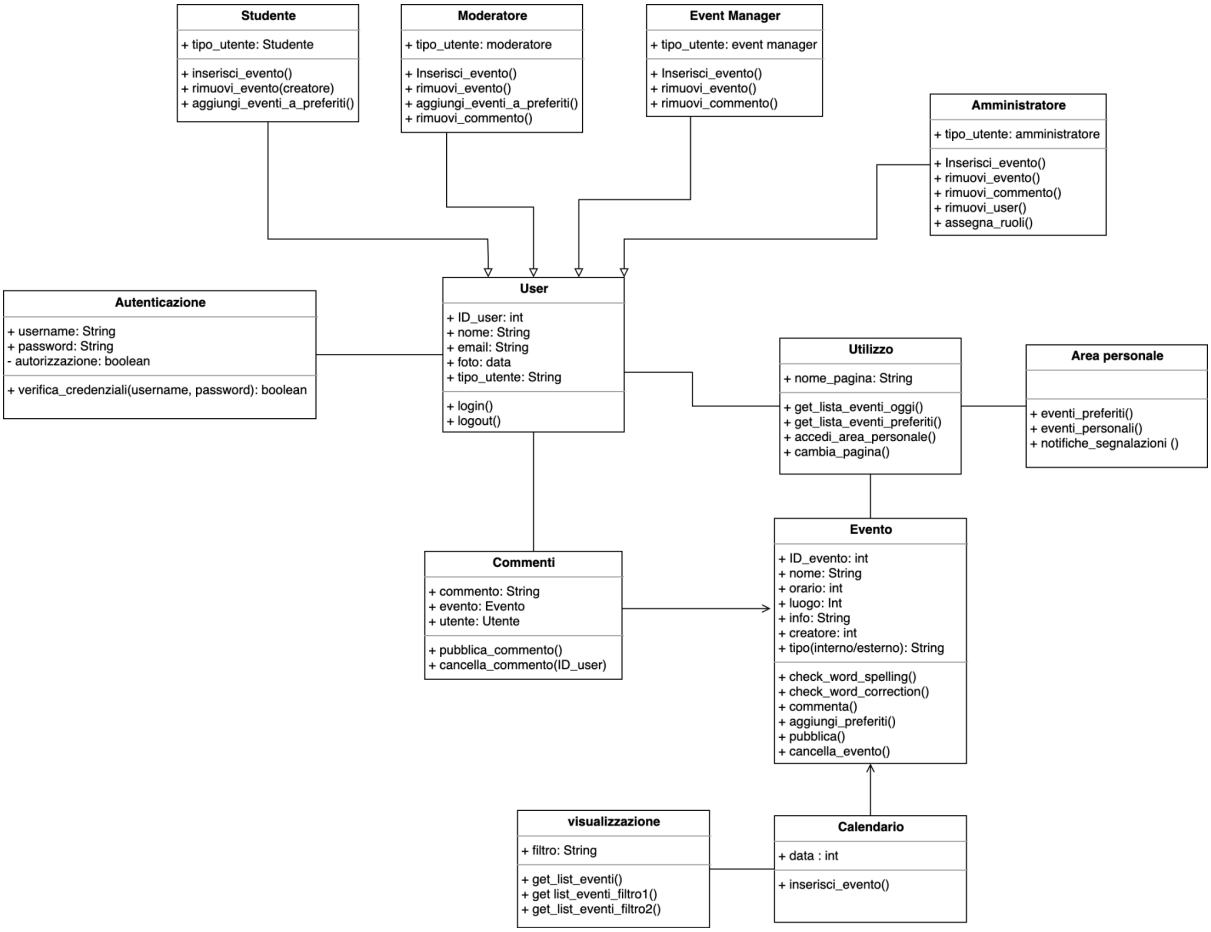
La classe *area personale* possiede la funzione di ‘eventi_preferiti()’, che restituisce la lista degli eventi a cui l’utente che la esegue ha aggiunto la stella.

La funzione di ‘eventi_personali()’ restituisce invece la lista degli eventi creati dall’utente in ordine cronologico di creazione.

La funzione di ‘notifiche_segnalazioni()’ è visibile soltanto ai moderatori, cioè agli utenti con la variabile ‘tipo_utente’ settata su moderatori. La funzione restituisce la lista delle segnalazioni di eventi e commenti in due riquadri separati. Per poi eliminare un evento/commento l’utente moderatore richiamerà una funzione interna alla sua classe.



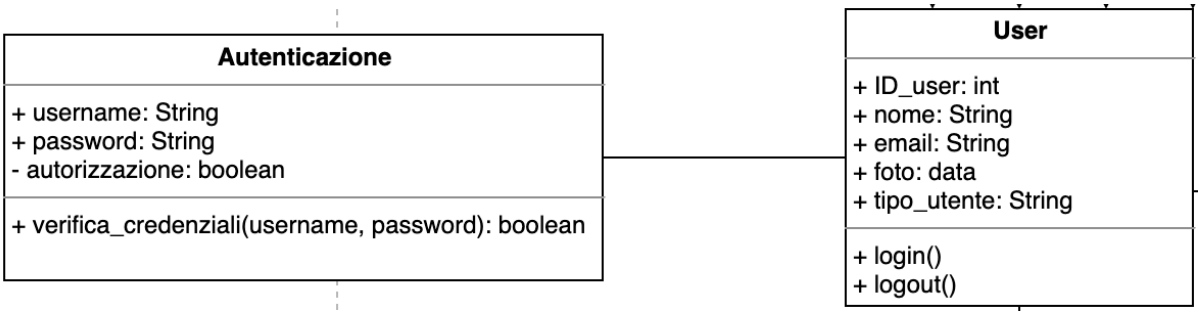
7 - Diagramma delle classi complessivo



OCL

1 - Login

Un utente è autorizzato al login solo quando l'attributo Autenticazione.autorizzazione è impostato a true dall'università.

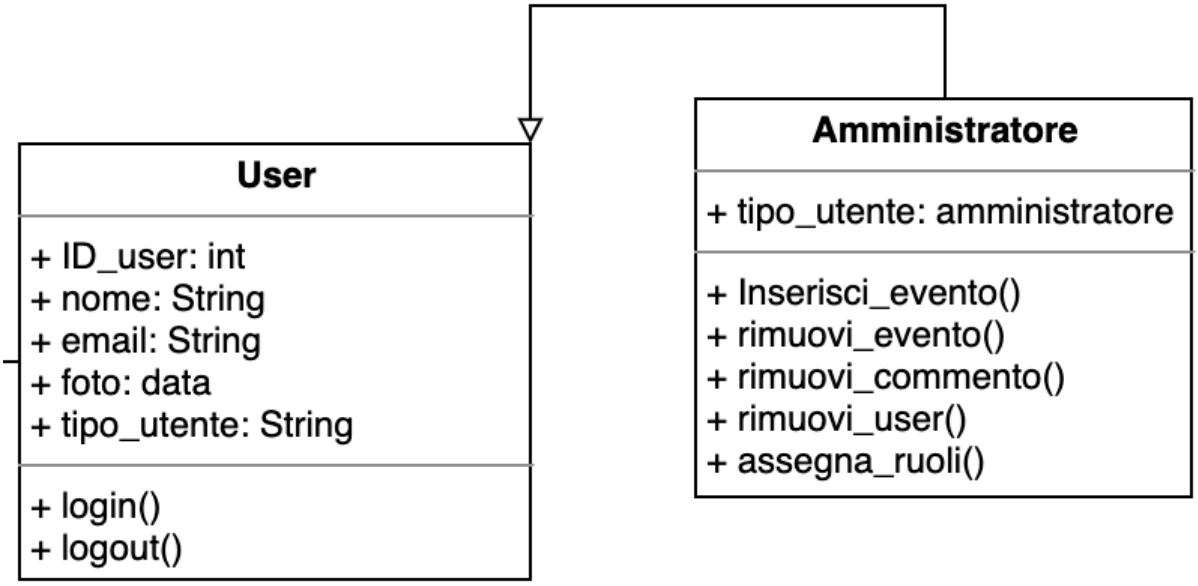


context Autenticazione::verifica_credenziali()
post: self.autorizzazione = true

context User::login()
pre: Autenticazione.autorizzazione = true

context User::logout()
post: Autenticazione.autorizzazione = false

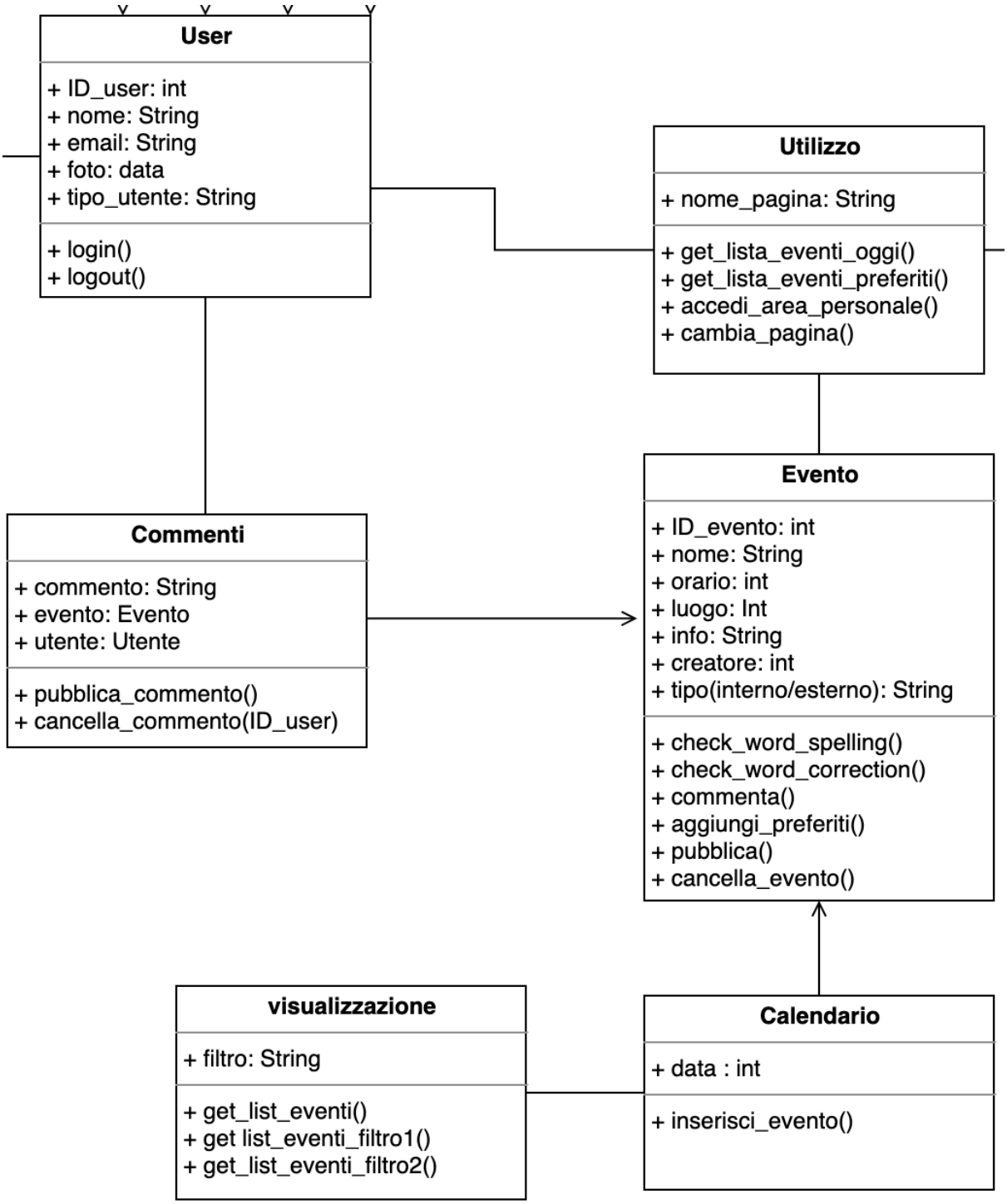
2 - Utenti amministratori di sistema




```
context Amministratore:: rimuovi_user()
post: User.ID_User = NULL

context Amministratore:: assegna_ruoli()
post: (User.tipo_utente = Studente)       OR
      (User.tipo_utente = Moderatore)     OR
      (User.tipo_utente = Event_manager)
```

3 - Gestione e visualizzazione eventi



3.1 - OCL per la gestione

```
context Evento:: pubblica()
pre: (Autenticazione.autorizzazione = true) AND
     (self.nome   != NULL)                  AND
     (self.orario != NULL)                 AND
     (self.luogo  != NULL)                  AND
     (self.orario >= Calendario.data)
post: (self.ID_evento != NULL) AND
      (self.creatore  != NULL)

context Evento:: commenta()
pre: (Autenticazione.autorizzazione = true)

context Evento:: cancella_evento()
pre: (User.tipo_utente = Event_manager) OR
     (User.ID_User = Evento.ID_Evento)  OR
     (User.tipo_utente = Moderatore)

context Commenti:: cancella_commento()
pre: (Autenticazione.autorizzazione = true) AND
     (self.utente = User.ID_user)
```

3.2 - OCL per la visualizzazione

```
context Utilizzo:: get_lista_eventi_oggi()
pre: Evento.orario = Calendario.data

context Visualizzazione:: get_list_eventi()
pre: Evento.orario >= Calendario.data
```

4 - Diagramma classi complessive con OCL