BeeLive
Deliverable 2

Gruppo 21:
Cipriani Pietro, 226959
Orlando Dennis, 227688
Ziviani Elia, 228172

22 Aprile 2024

Indice

1 Component Diagram

Questo capitolo è incentrato sull'analisi dei componenti del sistema che saranno realizzati.
Lo scopo è quello di descrivere ogni singolo componente in termini di funzionalità e interfacce.

I componenti sono individuati seguendo quando espresso nel documento Deliverable 1 per quanto riguarda i casi d'uso e sono definiti come entità autonome nel sistema.
Questi componenti sono provvisti di interfacce che permettono la comunicazione tra di essi.
Le interfacce sono definite come:

- **Interfacce di OUTPUT**: Tutte quelle interfacce che offrono servizi al sistema.
- **Interfacce di INPUT**: Tutte quelle interfacce che ricevono servizi dal sistema.

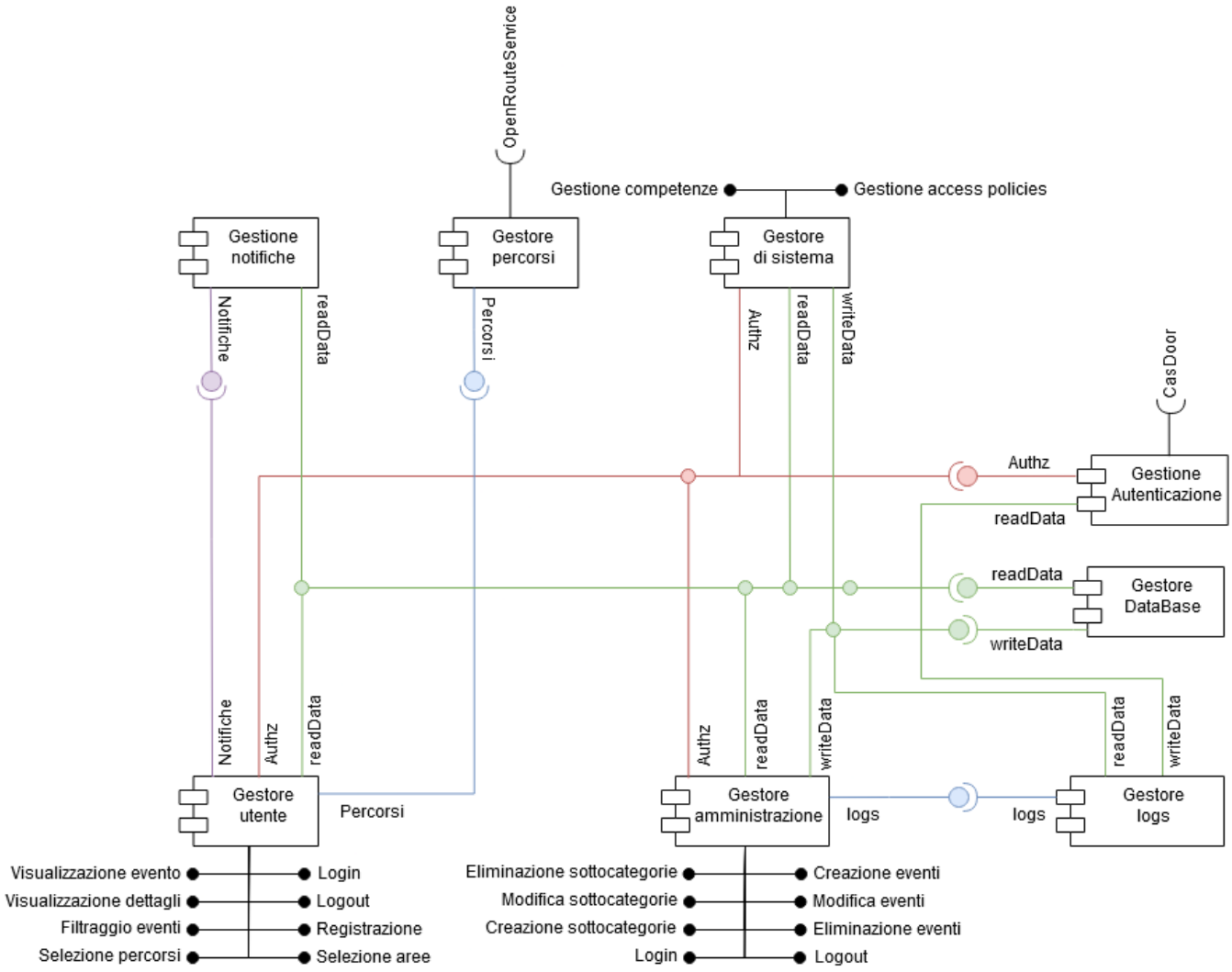

Figure 1: Diagramma dei componenti

1.1 Gestione utente

INPUT	Nome	Descrizione
	Notifiche	Il componente utilizza l'interfaccia fornita dal componente "Gestione Notifiche" per ricevere notifiche relative alle criticita'
	Auth	Il componente utilizza l'interfaccia fornita dal componente "Gestione Autenticazione" per richiedere il token necessario per effettuare il login
	Data	Il componente utilizza l'interfaccia fornita dal componente di gestione del database per scambiare tutti i dati necessari a garantire un corretto funzionamento dell'applicativo per gli utenti
	Percorsi	Il componente utilizza l'interfaccia fornita dal componente gestione percorsi per ottenere i percorsi calcolati evitando le criticità presenti in mappa
OUTPUT	**Nome**	**Descrizione**
	Login	Il componente fornisce un'interfaccia per effettuare il login attraverso i servizi del componente esterno CasDoor, che prevede l'inserimento delle specifiche credenziali per generare un token di accesso
	Logout	Il componente fornisce all'utente un'interfaccia per effettuare il logout
	Registrazione	Il componente fornisce all'utente un'interfaccia per registrarsi attraverso il servizio CasDoor
	Selezione aree	Il componente fornisce un'interfaccia che permette all'utente di specificare sulla mappa della citta' di Trento la sua zona di maggior interesse per poi considerare quella zona come utile per la visualizzazione dei dati
	Visualizzazione evento	Il componente fornisce un'interfaccia che permette la visualizzazione di tutti gli eventi presenti sul database riguardanti la citta' di Trento o la zona/il percorso d'interesse se specificati
	Visualizzazione dettagli	Il componente offre un'interfaccia per la visualizzazione di tutti i dettagli che caratterizzano il singolo evento selezionato
	Filtraggio eventi	E' offerta dal componente l'interfaccia che permette il filtraggio degli eventi tramite specificita' espresse dall'utilizzatore
	Selezione aree	Il componente offre l'interfaccia per la specifica delle aree di maggiore interesse secondo l'utente. Questa specifica e' utile per la visualizzazione delle sole criticita' contenute in essa
	Selezione percorsi	Il componente offre l'interfaccia per la specifica dei percorsi piu' significativi per l'utente permettendogli di essere notificato nel caso in cui nel percorso si verifichi una criticita'

Table 1: Tabella gestione utente

1.2 Gestione amministrazione

INPUT	Nome	Descrizione
	Auth	Il componente utilizza l'interfaccia fornita dal componente "Gestione Autenticazione" per richiedere il token necessario per effettuare il login
	Data	Il componente utilizza l'interfaccia fornita dal componente di gestione del database per scambiare tutti i dati necessari a garantire un corretto funzionamento dell'applicativo per l'amministrazione, come il salvataggio delle operazioni eseguite sugli eventi
OUTPUT	**Nome**	**Descrizione**
	Creazione eventi	Il componente offre internamente l'interfaccia per la creazione di un nuovo evento, seguendo e verificando che le informazioni utilizzate siano corrette
	Modifica eventi	Il componente offre internamente l'interfaccia utile alla modifica degli eventi gia' pubblicati e salvati in precedenza su database
	Eliminazione eventi	Il componente prevede internamente l'interfaccia che si occupa di eseguire le operazioni di eliminazione degli eventi pubblicati e salvati su database
	Creazione sottocategorie	Il componente offre internamente l'interfaccia che permette all'amministrazione di creare nuove sottocategorie degli eventi, fornendo una migliore organizzazione degli stessi
	Modifica sottocategorie	Il componente offre internamente l'interfaccia per eseguire modifiche sulle sottocategorie create e salvate su database
	Eliminazione sottocategorie	Il componente offre internamente l'interfaccia per permettere la cancellazione delle sottocategorie create e salvate su database in precedenza
	Data	Il componente utilizza questa interfaccia per eseguire il caricamento e il salvataggio dei nuovi dati di eventi e categorie su database

Table 2: Tabella gestione amministrazione

1.3 Gestione di sistema

INPUT	Nome	Descrizione
	Data	Il componente utilizza un'interfaccia fornita dal datalayer per accedere ai dati sul database
	Auth	Il componente utilizza l'interfaccia fornita dal componente "Gestione Autenticazione" per richiedere il token necessario per effettuare il login
OUTPUT	**Nome**	**Descrizione**
	Data	Il componente fornisce internamente un'interfaccia per accedere al database
	Gestione competenze	Il componente fornisce internamente un'interfaccia per la gestione delle aree di competenza degli utenti
	Gestione access policies	Il componente fornisce internamente un'interfaccia utile alla gestione delle access policies degli utenti nel sistema

Table 3: Tabella gestione di sistema

1.4 Gestione autenticazione

INPUT	Nome	Descrizione
	CasDoor	Il componente utilizza l'interfaccia fornita da CasDoor per l'autenticazione degli utenti. CasDoor e' un servizio specifico negli accessi degli utenti
	Data	Il componente utilizza l'interfaccia del componente "Gestione database" per autenticare gli utenti registrati
OUTPUT	**Nome**	**Descrizione**
	Auth	Il componente fornisce un'interfaccia agli altri componenti per usufruire del token generato all'accesso nei servizi previsti, in risposta all'inserimento dei dati accesso e alla loro validazione

Table 4: Tabella gestione autenticazione

1.5 Gestione notifiche

INPUT	Nome	Descrizione
	Data	Il componente utilizza un'interfaccia fornita dal datalayer per accedere ai dati sul database
OUTPUT	**Nome**	**Descrizione**
	Notifiche	Il componente fornisce internamente un'interfaccia per l'inoltro delle notifiche

Table 5: Tablla gestione notifiche

1.6 Gestione percorsi

INPUT	Nome	Descrizione
	Data	Il componente utilizza un'interfaccia fornita dal datalayer per accedere ai dati sul database
	OpenRouteService	Il componente utilizza un'interfaccia fornita dal servizio OpenRouteService per il calcolo dei percorsi
OUTPUT	**Nome**	**Descrizione**
	Percorsi	Il componente fornisce internamente un'interfaccia per la computazione dei percorsi che evitano le zone con criticita' di viabilita'

Table 6: Tabella gestione percorsi

1.7 Gestione database

INPUT	Nome	Descrizione
	Data	Il componente utilizza un'interfaccia fornita dal datalayer per accedere ai dati sul database
OUTPUT	**Nome**	**Descrizione**
	Data	Il componente fornisce internamente un'interfaccia per accedere al database

Table 7: Tabella gestione database

2 Diagramma delle classi

Nella sezione seguente verranno descritte nel dettaglio le classi che formeranno la struttura del sistema attraverso il diagramma delle classi.
Saranno quindi descritte, oltre alle classi, anche tutte le relazioni che intercorrono tra di esse e i ruoli dei gestori.

La descrizione sarà divisa, per motivi di chiarezza, in due sezioni. La prima tratterà le classi relative agli utenti, mentre la seconda tratterà le rimanenti.

2.1 Utenti

2.1.1 Utente

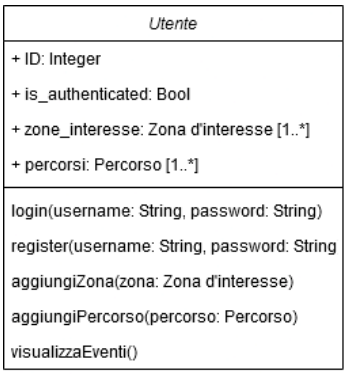

Figure 2: Classe Utente

Questa classe rappresenta la figura dell'utente che utilizza l'applciazione mobile senza seguire la procedura di autenticazione.
Come riportato nel primo documento "Deliverable 1" nella sezione relativa agli attori [Ref. 3.1.1, pag. 6], l'utente non autenticato ha accesso a tutte le funzionalità dell'applicazione salvando lo stato solamente sul dispositivo su cui è installata l'applicazione, quindi perdendo i dati una volta cambiato dispositivo.

Attributi:

Gli attributi che caratterizzano questa classe sono:

- **ID**: Attributo che rappresenta univocamente l'utente non autenticato.
- **is_authenticated**: Questo attributo definisce se l'utente è autenticato o meno. Per l'utente non autenticato il valore sarà "False".
- **zone_interesse**: Attributo che rappresenta la zona di interesse impostate dall'utente.
- **percorsi**: Attributo che rappresenta i percorsi che per l'utente risultano di interesse.

Metodi:

I metodi a disposizione della classe sono:

- **login(username, password)**: Metodo utilizzato per effettuare il login all'interno dell'applicazione. Utilizzando questo metodo l'utente diventa un "Utente autenticato".
  Sono utilizzati metodi del gestore "Gestione utente" e del gestore "Gestione autenticazione" per effettuare il login.
- **register(username, password)**: Metodo che permette all'utente di registrarsi per la prima volta all'interno dell'applicazione.
  Sono utilizzati metodi del gestore "Gestione utente" e del gestore "Gestione autenticazione" per effettuare la registrazione. Anche questo metodo fa diventare l'utente un "Utente autenticato".
- **aggiungiZona(zona)**: Metodo utilizzato dall'utente per selezionare la zona di interesse sulla mappa.
- **aggiungiPercorso(percorso):** Metodo che permette all'utente di selezionare il percorso di interesse sulla mappa.
- **visualizzaEventi()**: Metodo utilizzato per visualizzare gli eventi presenti sulla mappa o all'interno della zona d'interesse precedentemente specificata.

2.1.2 Utente autenticato

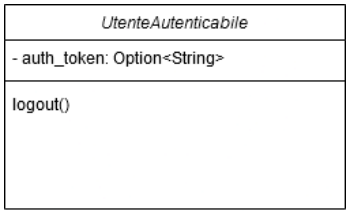

Figure 3: Classe Utente Autenticato

L'utente autenticato [Ref. 3.1.2, pag. 6] deriva le sue caratteristiche dall'utente non autenticato, ma ha la possibilità di salvare i dati sul server permettendo quindi di mantenere lo stato dell'applicazione anche cambiando dispositivo.

Attributi:
Sono gli stessi a disposizione dell'utente non autenticato, con l'aggiunta di:

- **auth_token**: Attributo che rappresenta il token di autenticazione dell'utente.

Metodi:
Anche i metodi sono gli stessi a disposizione dell'utente non autenticato, aggiungendo però il metodo:

- **logout()**: Metodo utilizzato per effettuare il logout dall'applicazione.
  Sono utilizzati metodi del gestore "Gestione utente" e del gestore "Gestione autenticazione" per effettuare il logout.

Ed eliminando la possibilita' di richiamare i metodi "login" e "register", superflui in quanto l'utente è già autenticato.

2.1.3 Utente amministratore

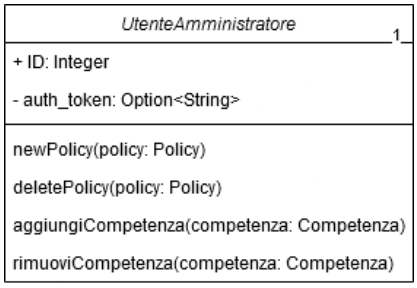

Figure 4: Classe Utente Amministratore

L'utente amministratore [Ref. 3.1.4, pag. 6] è la classe che rappresenta l'utente che ha il controllo totale sul sistema, con la possibilita' di definire competenze e policy di accesso per gli utenti.

Attributi:

Gli attributi che caratterizzano questa classe sono:

- **ID**: Attributo che identifica univocamente l'utente amministratore.
- **auth_token**: Attributo che rappresenta il token di autenticazione dell'utente.

Metodi:

I metodi a disposizione della classe sono:

- **login(username, password)**: Metodo utilizzato per effettuare il login all'interno dell'applicazione. Utilizzando questo metodo l'utente diventa un "Utente autenticato".
  Sono utilizzati metodi del gestore "Gestione utente" e del gestore "Gestione autenticazione" per effettuare il login.
- **logout(username, password)**: Metodo utilizzato per effettuare il logout dall'applicazione, cosi' da disconnettere l'utente dal servizio
- **aggiungiPolicy(policy)**: Metodo che permette all'utente amministratore di definire le policy di accesso per gli utenti.
- **rimuoviPolicy(policy)**: Metodo che permette all'utente amministratore di rimuovere le policy di accesso per gli utenti.
- **aggiungiCompetenza(competenza)**: Metodo che permette all'utente amministratore di definire le competenze degli utenti.
- **rimuoviCompetenza(competenza)**: Metodo che permette all'utente amministratore di rimuovere le competenze degli utenti.

2.1.4 Utente autorizzato

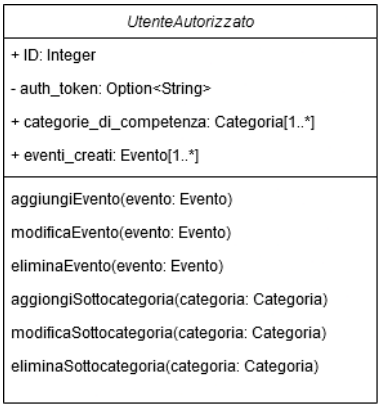

Figure 5: Classe Utente Autorizzato

L'utente autorizzato [Ref. 3.1.3, pag. 6] è la classe che rappresenta tutte le entita' comunali che hanno i permessi di inserire gli eventi a sistema.
Questi utenti accedono al sistema attraverso l'applicativo desktop, sviluppato apposta per la gestione degli eventi.

Attributi:
Gli attributi che definiscono questa classe sono:

- **ID**: Caratteristica che identifica univocamente l'utente autorizzato nel sistema.
- **auth_token**: Attributo che rappresenta il token di autenticazione dell'utente.
- **categorie_di_competenza**: Riporta le categorie su cui l'utente ha competenza.
- **eventi_creati**: Tutti gli eventi che l'utente ha creato e salvato sul database.

Metodi:
I metodi a disposizione della classe sono:

- **login(username, password)**: Metodo utilizzato per effettuare il login all'interno dell'applicazione. Utilizzando questo metodo l'utente diventa un "Utente autenticato".
  Sono utilizzati metodi del gestore "Gestione utente" e del gestore "Gestione autenticazione" per effettuare il login.
- **logout(username, password)**: Metodo utilizzato per effettuare il logout dall'applicazione, cosi' da disconnettere l'utente dal servizio
- **aggiungiEvento(evento)**: Metodo che permette all'utente autorizzato di creare ed aggiungere un evento al sistema.
- **modificaEvento(evento)**: Metodo che permette all'utente autorizzato di modificare ed aggiornare un evento precedentemente creato.
- **eliminaEvento(evento)**: Metodo che permette all'utente autorizzato di eliminare dal sistema un evento precedentemente creato.
- **aggiongiSottocategoria(categoria)**: Metodo che permette all'utente autorizzato di creare ed aggiungere una sottocategoria di eventi al sistema.
- **modificaSottocategoria(categoria)**: Metodo che permette all'utente autorizzato di modificare ed aggiornare una sottocategoria di eventi precedentemente creata.
- **eliminaSottocategoria(categoria)**: Metodo che permette all'utente autorizzato di eliminare dal sistema una sottocategoria di eventi precedentemente creata.

2.2 Altre classi

2.2.1 Evento

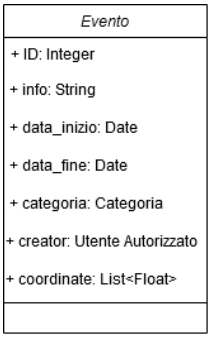

Figure 6: Classe Evento

La classe "Evento" rappresenta un evento o criticita' che l'utente amministratore pubblica sul sistema e che e' possibile visualizzare da applicativo mobile.

Attributi:

- **ID**: Attributo che identifica univocamente l'evento pubblicato sul sistema.
- **titolo**: Attributo che rappresenta il titolo dell'evento.
- **descrizione**: Attributo che rappresenta la descrizione dell'evento.
- **data_inizio**: Attributo che riporta la data d'inizio dell'evento.
- **data_inizio**: Attributo che riporta la data di fine dell'evento.
- **categoria**: Attributo che rappresenta la categoria a cui l'evento appartiene.
- **creator**: Attributo che rappresenta l'utente autorizzato che ha creato l'evento.
- **coordinate**: Attributo che rappresenta il luogo in cui l'evento si svolgera'.

Per la classe in questione non sono previsti metodi.

2.2.2 Percorso

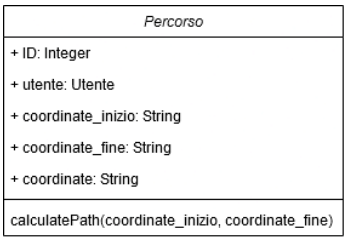

Figure 7: Classe Percorso

La classe "Percorso" rappresenta un percorso che l'utente richiede di calcolare da un punto d'inizio ad un punto di fine.
Il percorso che risulta rappresenta la via per raggiungere il punto di destinazione evitando tutte le criticita' presenti su mappa.

Attributi:
Gli attributi che caratterizzano questa classe sono:

- **ID**: Attributo che identifica univocamente il percorso richiesto dall'utente.
- **utente**: Attributo che rappresenta univocamente l'utente che ha richiesto il percorso.
- **coordinate_inizio**: Attributo che rappresenta le coordinate del punto di partenza del percorso.
- **coordinate_fine**: Attributo che rappresenta le coordinate del punto di arrivo del percorso.
- **coordinate**: Attributo che rappresenta tutte le coordinate tra il punto di partenza e di arrivo del percorso, una volta che e' stato calcolato per evitare situazioni critiche su mappa.

Metodi:
Il metodo a disposizione della classe è:

- **calcolaPercorso(coordinate_inizio, coordinate_fine)**: Metodo invocato per eseguire il calcolo del percorso dal punto di partenza al punto di arrivo evitando tutte le criticita' presenti su mappa.

2.2.3 Zone di interesse

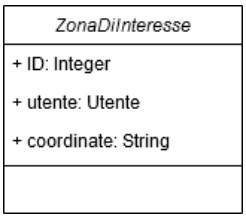

Figure 8: Classe Zona di Interesse

La classe "Zone di interesse" rappresenta le zone che l'utente ha selezionato come di interesse per la visualizzazione degli eventi e delle criticita' presenti su mappa.

Attributi:
Gli attributi che caratterizzano questa classe sono:

- **ID**: Attributo che identifica univocamente la zona di interesse selezionata dall'utente.
- **utente**: Attributo che rappresenta univocamente l'utente che ha selezionato la zona di interesse.
- **coordinate**: Attributo che rappresenta le coordinate della zona di interesse selezionata dall'utente.

Per la classe in questione non sono previsti metodi.

2.2.4 Categoria

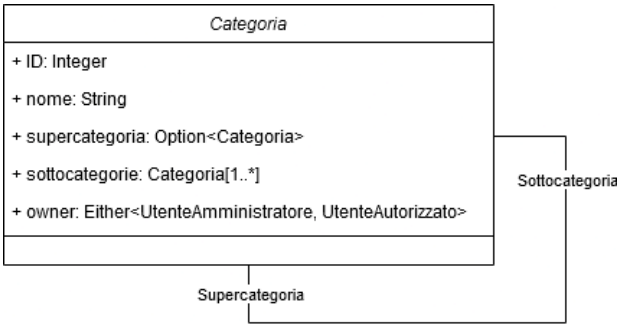

Figure 9: Classe Categoria

La classe "Categoria" rappresenta le categorie di eventi presenti sul sistema, definibili dagli utenti autorizzati dal comune di Trento.

Attributi:
Gli attributi che caratterizzano questa classe sono:

- **ID**: Attributo che identifica univocamente la categoria di eventi.
- **nome**: Attributo che rappresenta il nome della categoria di eventi.
- **supercategoria**: Attributo che rappresenta la supercategoria a cui la categoria appartiene.
- **sottocategorie**: Attributo che rappresenta le sottocategorie di eventi presenti nella categoria.
- **owner**: Attributo che rappresenta l'utente autorizzato che ha creato la categoria.

Per la classe in questione non sono previsti metodi.

2.2.5 Competenza

La classe "Competenza" rappresenta le competenze degli utenti autorizzati per la gestione degli eventi.

Attributi:
Gli attributi che caratterizzano questa classe sono:

- **ID**: Attributo che identifica univocamente la competenza dell'utente.
- **id_utente**: Attributo che rappresenta univocamente l'utente autorizzato che ha la competenza.

Da definire

2.2.6 Policy

La classe "Policy" rappresenta le policy di accesso degli utenti al sistema.

Attributi:
Gli attributi che caratterizzano questa classe sono:

- **ID**: Attributo che identifica univocamente la policy di accesso.
- **nome**: Attributo che rappresenta il nome della policy di accesso.
- **utenti**: Attributo che rappresenta gli utenti autorizzati che seguono la policy.

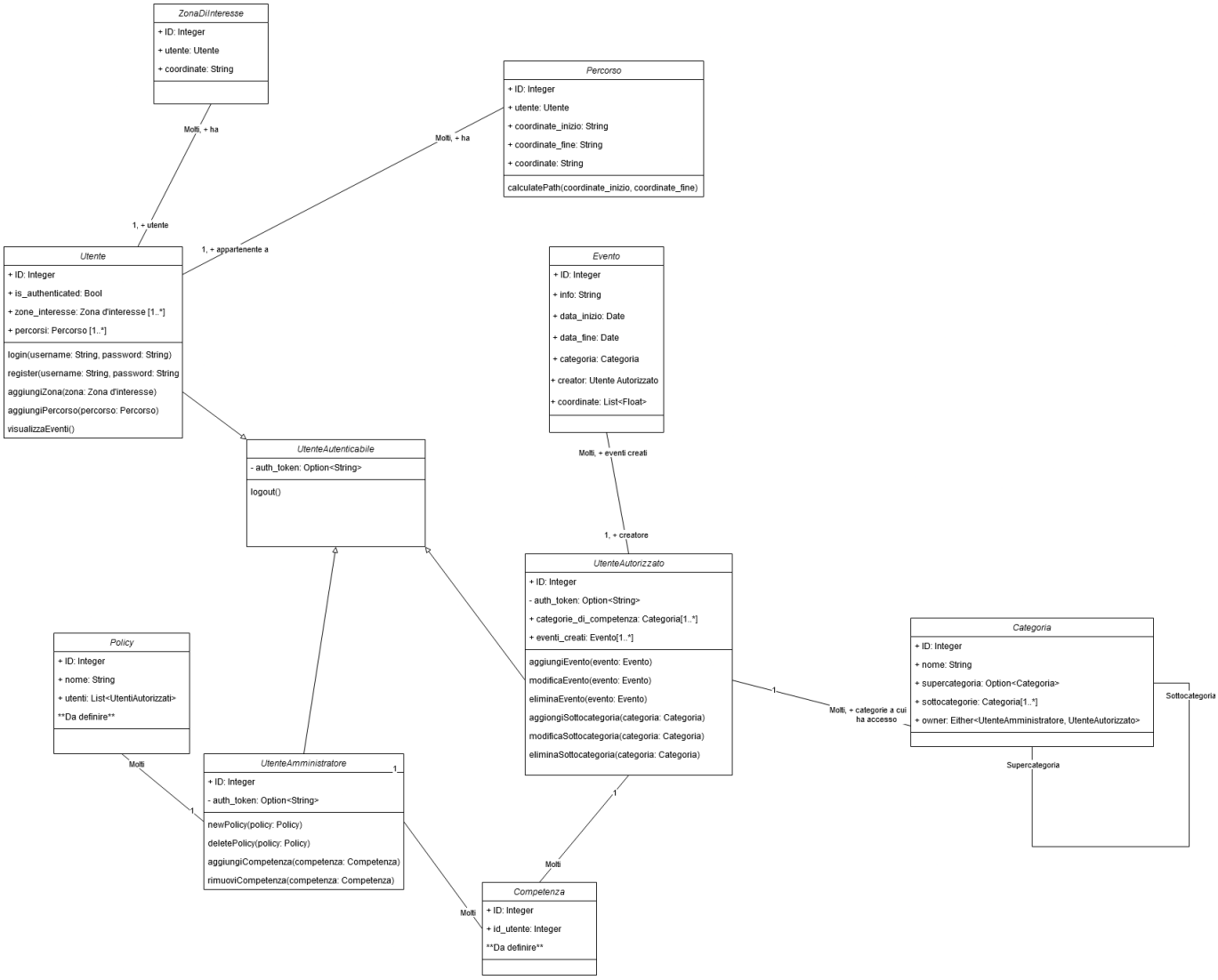

Figure 10: Diagramma delle classi

3 Contstraints con OCL

3.1 Utente

```
context Utente
    inv: self.zone_interesse -> forAll(z | z.utente == self)
    inv: self.percorsi -> forAll(p | p.utente == self)
    inv: self.auth_token.isEmpty() == self.is_authenticated
```

3.2 Utente autenticabile

```
context UtenteAutenticabile::login(_username, _password)
    pre: self.is_authenticated == false
    post: !self.auth_token.isEmpty()

context UtenteAutenticabile::deauthenticate()
    pre: self.is_authenticated == true
    post: self.auth_token.isEmpty()

context Utente::aggiungiZona(zona: ZonaDiInteresse)
    post: self.zone_interesse -> include(zona)

context Utente::aggiungiPercorso(percorso: Percorso)
    post: self.percorsi -> include(percorso)
```

3.3 Evento

```
context Evento
    inv: self.creator.categorie_di_competenza -> includes(self.categoria)
    inv: self.creator.eventi_creati -> includes(self)
    inv: self.data_inizio <= self.data_fine
    inv: !self.info.isEmpty()
```

3.4 Categoria

```
# l'Admin può creare solo categorie "root" con profondità massima di 1
    inv: if instanceof(self.owner) == UtenteAmministratore:
        self.supercategoria == None || self.supecategoria->forAll(s | s.supercategoria == None)

# ogni sottoalbero di categorie deve avere una radice in un nodo "Root"
```

3.5 Utente Autorizzato

```
context UtenteAutorizzato
    inv: self.eventi_creati -> forAll(e | e.creator == self)
    inv: self.categorie_di_competenza -> forAll()

context UtenteAutorizzato::aggiungiEvento(evento: Evento)
    post: self.eventi_creati -> include(evento)

context UtenteAutorizzato::aggiungiSottocategoria(categoria: Categoria)
    pre: !categoria.supercategoria.isEmpty()
    pre: self.categorie_di_competenza -> includes(categoria.supercategoria)
    post: self.categorie_di_competenza -> includes(categoria)
    post: categoria.owner == self

context UtenteAutorizzato::eliminaSottocategoria(categoria: Categoria)
    pre: self.categorie_di_competenza -> includes(categoria)
    pre: instanceof(categoria.owner) != UtenteAutorizzato
    post: self.categorie_di_competenza -> !includes(categoria)

context UtenteAutorizzato::modificaSottocategoria(categoria: Categoria)
    pre: self.categorie_di_competenza -> includes(categoria)
    pre: instanceof(categoria.owner) != UtenteAmministratore
    post: self.categorie_di_competenza -> includes(categoria.supercategoria)
    post: categoria.owner == categoria@pre.owner

context UtenteAutorizzato::aggiungiEvento(evento: Evento)
    post: self.categorie_di_competenza -> includes(evento.categoria)
    post: self.eventi_creati -> includes(evento)
    post: evento.creator == self

context UtenteAutorizzato::modificaEvento(evento: Evento)
    pre: evento.creator == self
    post: evento.creator == self

context UtenteAutorizzato::eliminaEvento(evento: Evento)
    pre: evento.creator == self
    post: self.eventi_creati -> !includes(evento)
```