Versione 1

Ingegneria del Software - DISI - Anno Accademico 2023-2024

Daniele Ye - Alessandro Dalfovo. - Giovanni Panighel

Nome progetto:

Tripply

Gruppo G42 - Deliverable 3

Scopo del documento

Il presente documento riporta la definizione dell'architettura del progetto ACCESS light 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 ACCESS light. 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 Gestione utenti e profilo

Secondo il diagramma di contesto di questo progetto, si può notare la presenza di un attore, **Trippler**, che caratterizza l'utente comune che utilizza i servizi comuni a tutti dell'applicativo web. Trippler estende la classe Utente, che contiene attributi che descrivono l'utente, per questo motivo è presente la pagina gestione profilo.

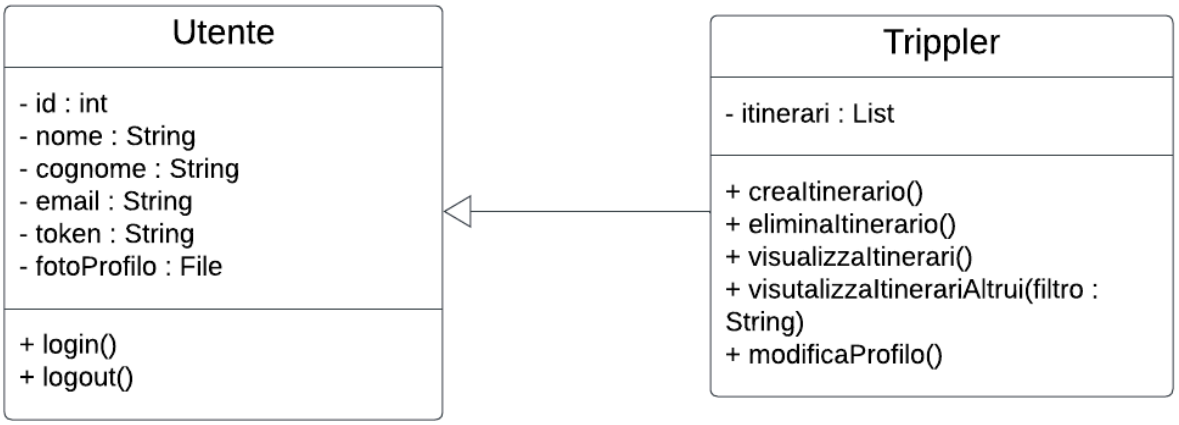


Figura 1. Classi per utenti

1.2 Gestione autenticazione

Il diagramma di contesto presenta un componente per la gestione dell'autenticazione. Questo sistema permette, come dice il nome, al singolo utente di autenticarsi al sistema. Sono stati predisposti 3 sotto sistemi per l'autenticazione: Auth0, Facebook e Google.

- Auth0 è un servizio esterno che permette di autenticarsi tramite una mail e password
- Google è un famoso social dove viene delegata la gestione dell'account
- Facebook, anche questo è un famoso social dove viene delegata anche qui la gestione dell'account

Tutti questi sistemi devono implementare un metodo autentica che in base al sistema utilizzato risponde con il token in caso di autenticazione avvenuta con successo, altrimenti risponde con un messaggio di errore.

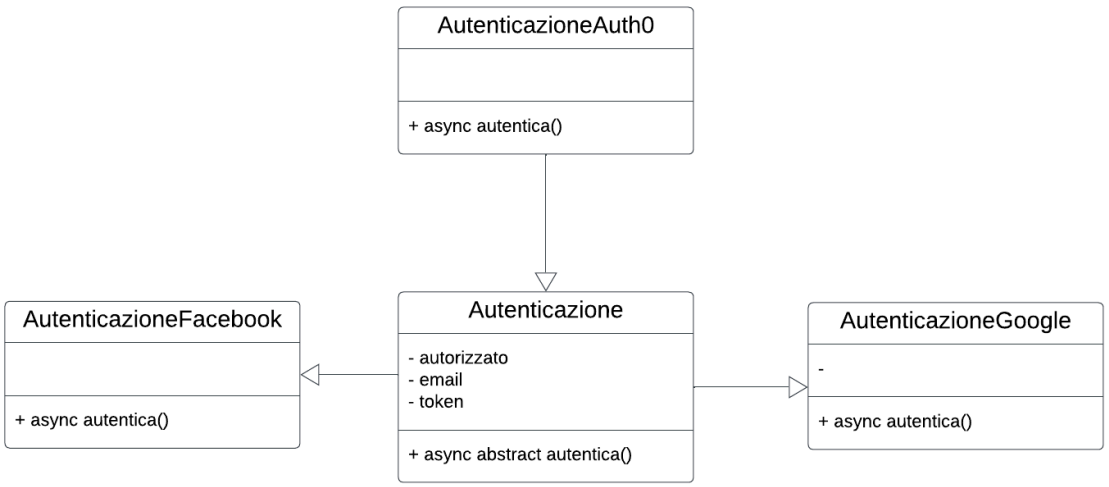


Figura 2. Classi per la gestione dell'autenticazione

1.3 Gestione itinerari

Analizzando i componenti, Creazione Itinerario, Creazione Giorno e Creazione Tappa, è stato definito il seguente diagramma delle classi, Itinerario, Giorno e Tappa. In questo modo l'utente ha la possibilità di suddividere un itinerario in più giorno, ogni giorno formato da una o più tappe. Di seguito lo schema delle classi.

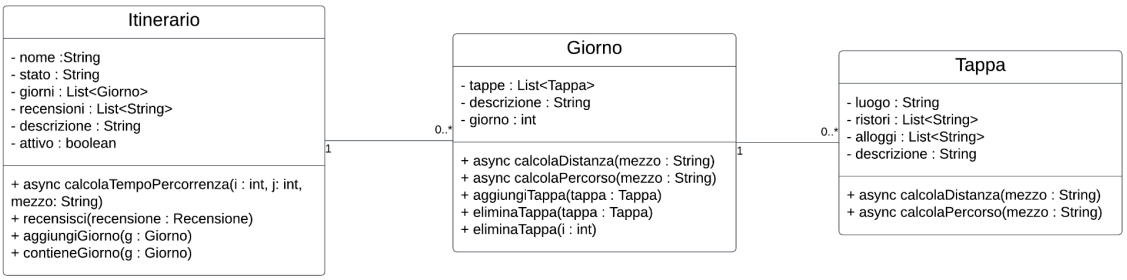


Figura 3. Classi per la gestione degli itinerari

1.4 Visualizzazione itinerari della community

Per il componente "Pagina Visualizzazione Itinerari Community" verrà creata una pagina dove sono presenti tutti gli itinerari presenti nel sistema, raggruppati per pagine. Per una maggiore fruizione del servizio è stato scelto di aggiungere dei filtri alla ricerca, come filtro per nome, per Stato e per durata. Questo permetterà all'utente di cercare esattamente quello di cui necessita.

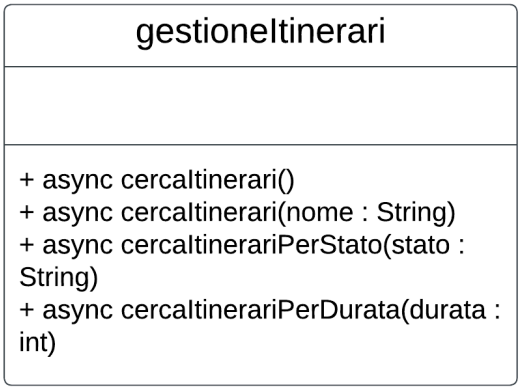


Figura 4. Classi per la gestione degli itinerari della community

1.5 Gestione delle recensioni

Il componente “Gestione delle recensioni” prevede la creazione di un sistema dove è possibile recensire un itinerario con un descrizione (testo) e un voto numerico. Questo permette inoltre, collegandosi al componente di prima, di ordinare la lista degli itinerari in base alla media dei voti delle recensioni.

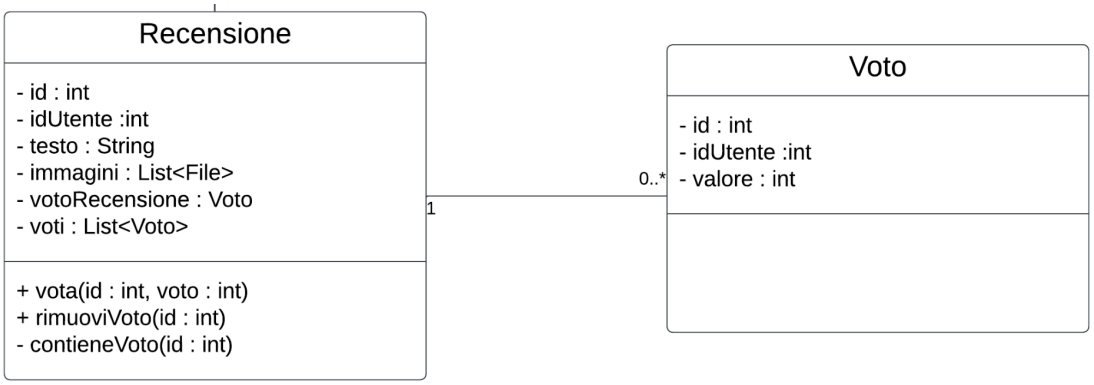


Figura 5. Classi per la gestione del sistema di recensioni

1.6 Diagramma Gestore Itinerari Salvati

Il componente Gestore Itinerari Salvati permette di aggiungere e rimuovere dalla lista di itinerari salvati dell'utente gli itinerari di altri utenti.

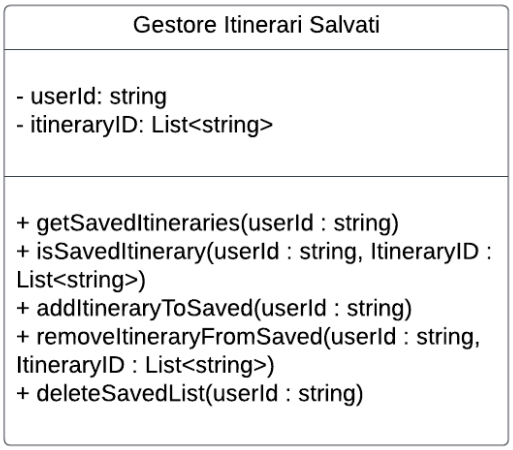
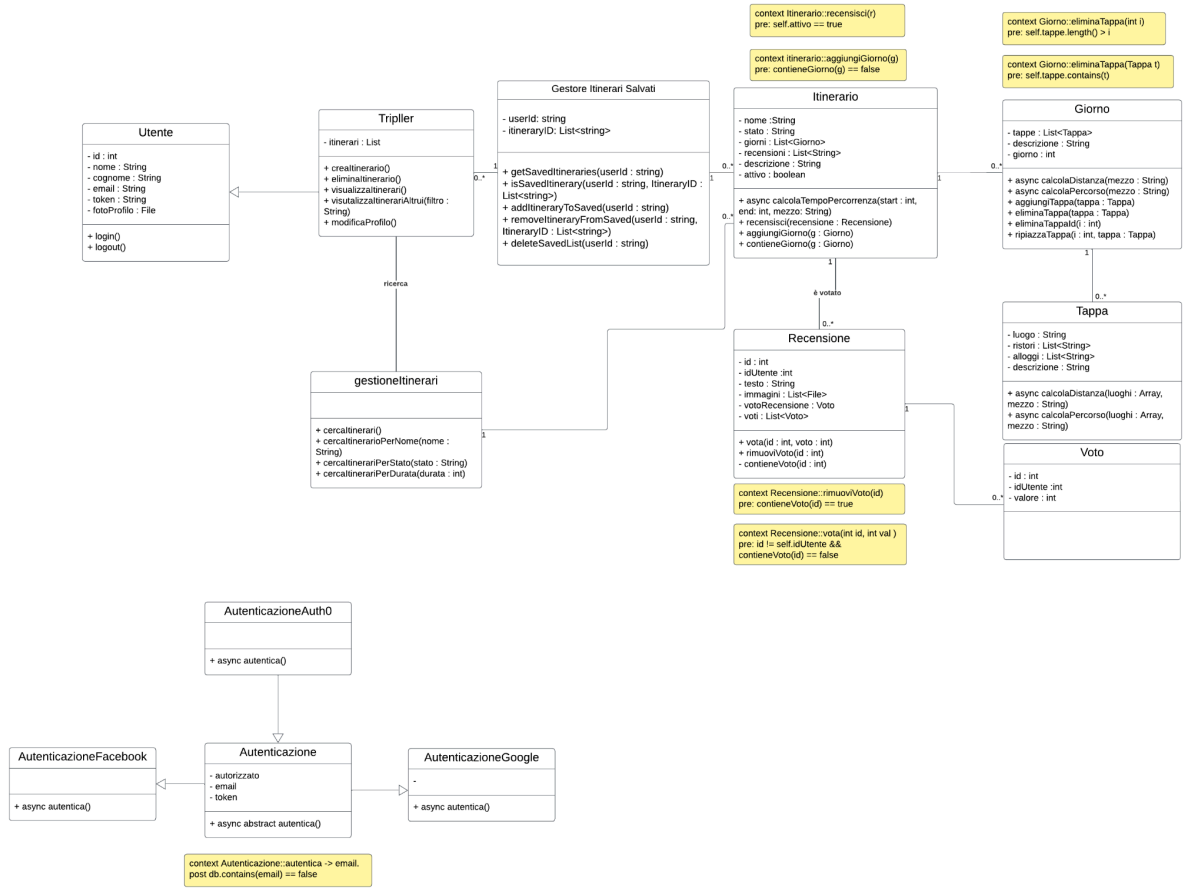


Figura 5. Classi per la gestione itinerari salvati

1.7 Diagramma delle classi complessivo

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



presentate.

Figura 6. Diagramma delle classi complessivo

2. Codice in Object Constraint Language

In questo capitolo viene formalmente delineata la logica prevista per determinate operazioni di alcune classi. Si adotta l'Object Constraint Language (OCL) poiché tali concetti non possono essere espressi in alcun altro modo formale all'interno del contesto di UML.

2.1 Registrazione

La registrazione al sistema può avvenire solamente se la mail con la quale ci si sta registrando non è già presente nel database.

Condizione sulle classi della figura 2.

Espressa in OCL attraverso una postcondizione con questo codice:

```
context Autenticazione::autentica -> email.
post db.contains(email) == false
```

2.2 Recensire un itinerario

La creazione di una recensione per un determinato itinerario è concessa solamente se non è presente un'altra recensione dallo stesso utente per lo stesso itinerario.

Condizione sulla classe:

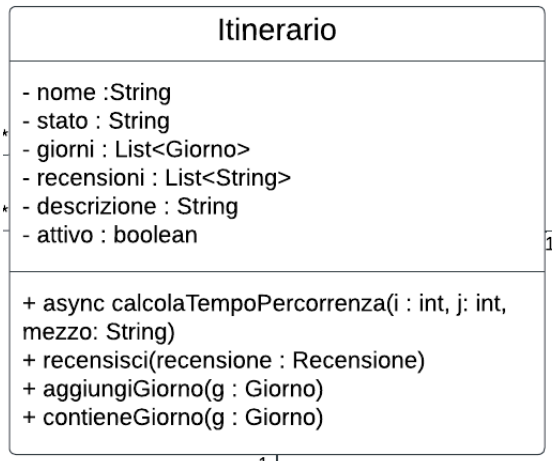


Figura 7. Classe Itinerario

Espressa in OCL attraverso una precondizione con questo codice:

```
context Itinerario::recensisci(r)
pre: self.attivo == true
```

2.3 Aggiunta di un giorno ad un itinerario

L'aggiunta di un giorno ad un itinerario è possibile solo nel caso quel giorno non sia già presente all'interno della lista dei giorni dell'itinerario.

Condizione sulla classe precedente.

Espressa in OCL attraverso una precondizione con questo codice:

```
context itinerario::aggiungiGiorno(g)
pre: contieneGiorno(g) == false
```

2.4 Eliminazione dell'ennesima tappa

E' possibile eliminare una tappa ad un indice definito, solo se la lista delle tappe è formata da almeno indice + 1 tappe.

Condizione sulla classe:

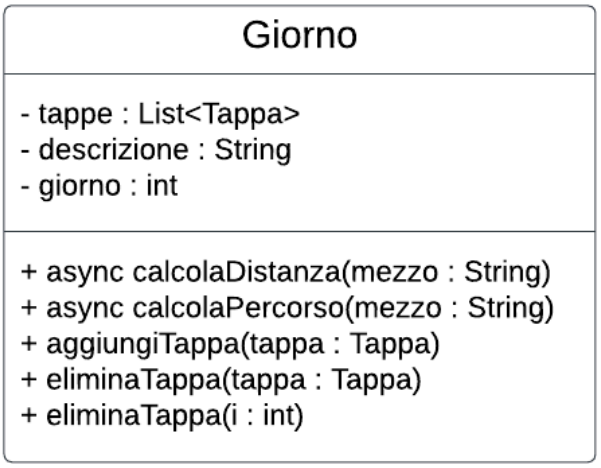


Figura 8. Classe Giorno

Espressa in OCL attraverso una precondizione con questo codice:

```
context Giorno::eliminaTappa(int i)
pre: self.tappe.length() > i
```

2.5 Eliminazione di una tappa specifica

E' possibile eliminare una tappa per riferimento solamente se quella tappa è presente nella lista delle tappe.

Condizione sulla classe precedente.

Espressa in OCL attraverso una precondizione con questo codice:

```
context Giorno::eliminaTappa(Tappa t)
pre: self.tappe.contains(t)
```

2.6 Rimuovere il voto di una recensione

Poiché una recensione può essere votata a sua volta, deve essere controllato se è possibile rimuovere il voto. L'azione può essere eseguita se la lista di voti contiene il voto dell'utente.

Condizione sulla classe:

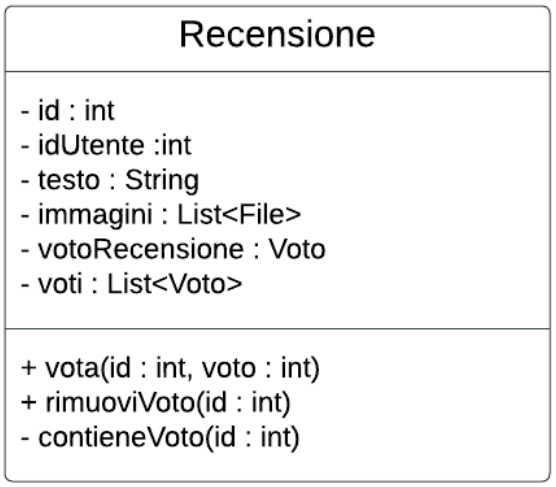


Figura 9. Classe Recensione

Espressa in OCL attraverso una precondizione con questo codice:

```
context Recensione::rimuoviVoto(id)
pre: contieneVoto(id) == true
```

2.7 Aggiunta di un voto ad una recensione

L'aggiunta di un voto ad una recensione è consentita solo se l'utente che sta votando non è il creatore della recensione e se l'utente non ha già votato la recensione.

Condizione sulla classe precedente.

Espressa in OCL attraverso una precondizione con questo codice:

```
context Recensione::vota(int id, int val )
pre: id != self.idUtente &&
contieneVoto(id) == false
```

3. Codice in Object Constraint Language

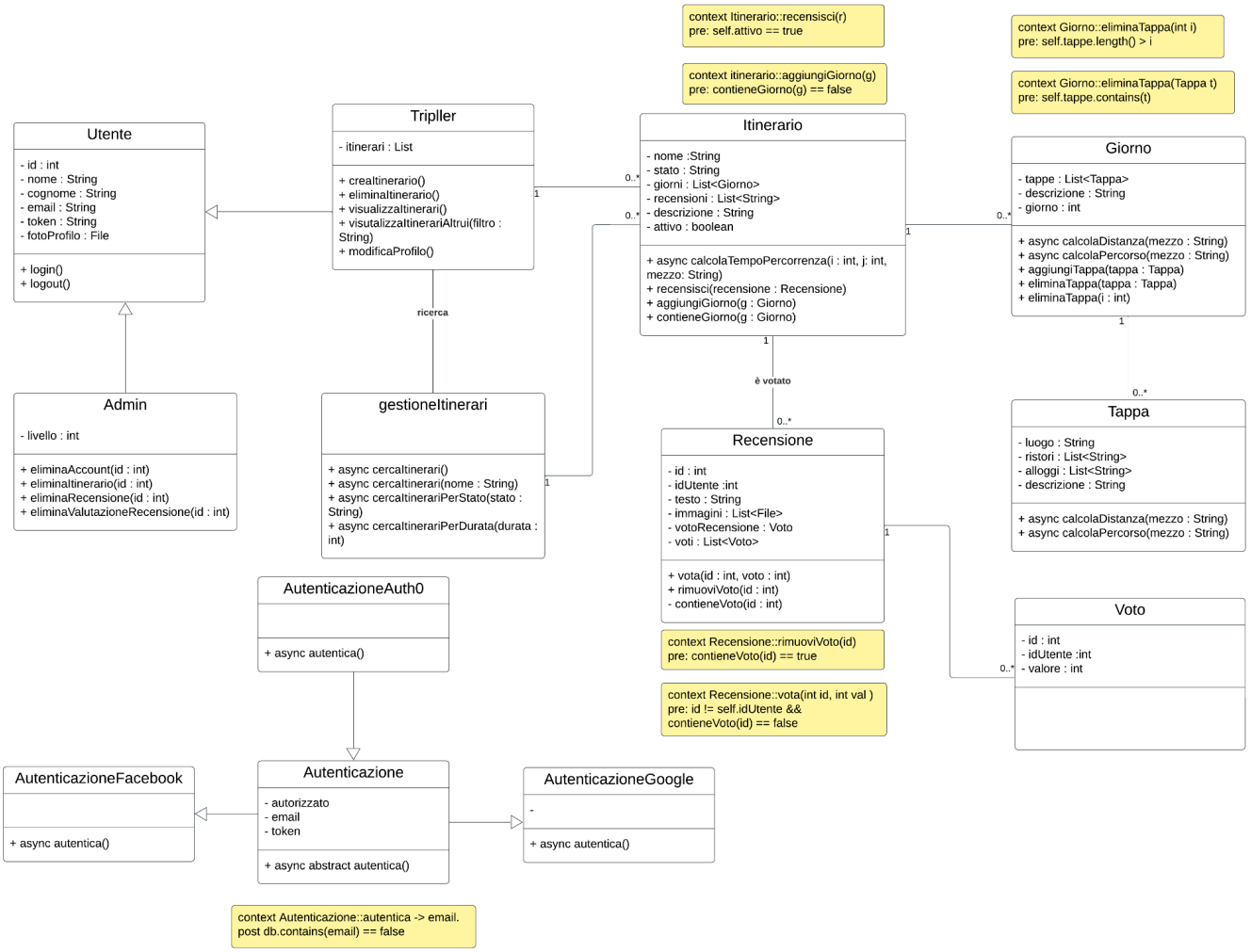


Figura 10. Diagramma delle classi con OCL

link all'immagine ad alta risoluzione qui