UNIVERSITÀ DEGLI STUDI DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Deliverable 3: Documento di architettura

Gruppo T16:
- Hossain Rafiu
- Missagia Fabio
- Posenato Alessandro

Sommario

1. Scopo del documento

In questo documento viene presentata la definizione dell'architettura del progetto HouseFinder mediante l'utilizzo di diagrammi delle classi UML e codice OCL. In precedenza, sono stati presentati il diagramma degli use case, il diagramma di contesto e quello dei componenti. Tenendo conto di questa progettazione, viene ora definita l'architettura del sistema, specificando sia le classi che dovranno essere implementate a livello di codice, rappresentate mediante un diagramma delle classi UML, sia la logica che regola il comportamento del software, descritta in OCL poiché tali concetti non possono essere espressi in altro modo all'interno del contesto UML.

2. Diagramma delle classi

Nel presente capitolo vengono presentate le classi previste per il sistema HouseFinder. Ogni componente presente nel diagramma dei componenti viene rappresentato da 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. Di seguito sono riportate le classi individuate a partire dai diagrammi di contesto e dei componenti.

2.1. Utente

Analizzando il diagramma di contesto realizzato per il progetto HouseFinder si nota la presenza di due attori “**Utente Autenticato**” e “**Utente Anonimo**”.

L’ attore “**Utente Anonimo**” è colui che utilizza il sito per la sola ricerca e visualizzazione degli annunci. L’ “Utente Anonimo” può effettuare il login o la registrazione per acquisire ulteriori funzionalità.

Infatti, una volta eseguita l’autenticazione, esso diventa “**Utente Autenticato**” e può procedere anche a pubblicare annunci, scrivere un messaggio ad altri utenti o salvare annunci e ricerche.

L’”Utente Anonimo” può anche effettuare il recupero della password, nel caso di dimenticanza.

Inoltre l’“Utente Autenticato" presenta vari metodi che riguardano la modifica dei dati del proprio account, come per esempio l’email e la password.

Entrambi questi attori hanno specifiche funzioni e attributi ma hanno anche molto in comune.
Sono state quindi individuate le due classi “**Utente Anonimo**” e “**Utente Autenticato**” con funzioni e attributi specifici e una classe “**Utente Generico**” con funzioni e attributi in comune collegate tramite una generalizzazione.

Nella classe “Utente Autenticato” sono state aggiunte anche le operazioni verso il sistema esterno “Metodo di Pagamento”, anch’esso presente nel diagramma di contesto.

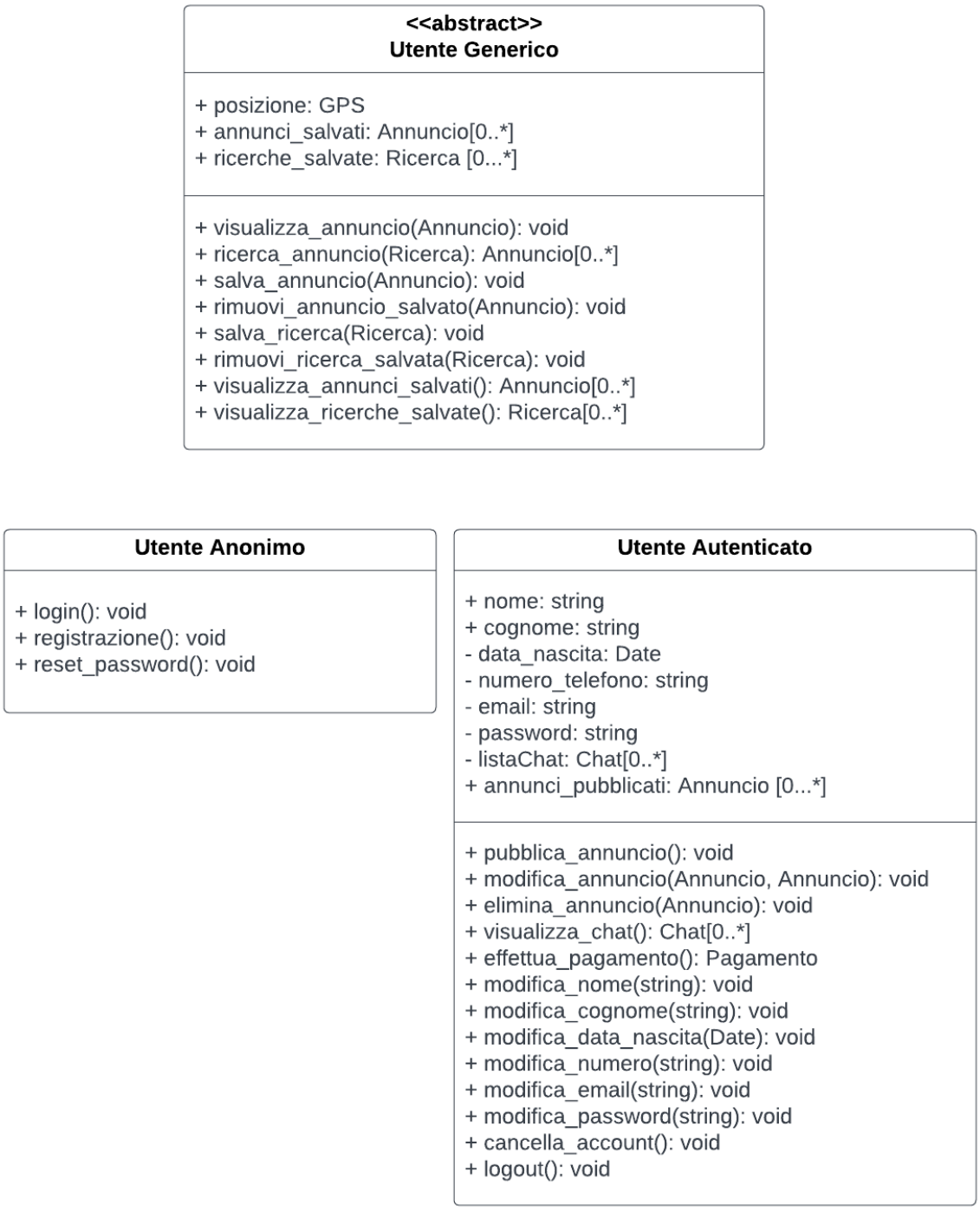

Figura 1. Classi per utenti e sistemi esterni

2.2. Autenticazione

Il diagramma di contesto analizzato presenta un sistema subordinato denominato “Google OAuth”. Questo elemento rappresenta il meccanismo di autenticazione degli utenti attraverso un sistema esterno di gestione delle credenziali proprietario di Google. È stata quindi identificata una classe “**Autenticazione**”, che si interfaccerà con questo sistema di gestione delle credenziali.

Ciò nonostante, il software sviluppato per il progetto HouseFinder gestirà e memorizzerà ugualmente username e password inserite dall’utente, visto che l’accesso con l’account Google è facoltativo.
Di conseguenza, il sistema di autenticazione utilizzerà sia il proprio sistema interno per la gestione delle credenziali degli utenti, sia il servizio di autenticazione di terze parti Google OAuth, per gli utenti che desiderano effettuare il login tramite Google.

Nel secondo caso il software passerà i dati di autenticazione al sistema esterno che valuterà questi dati e risponderà specificando se le credenziali inserite sono valide o meno.
Di seguito il dettaglio di questa classe con i propri attributi e metodi.

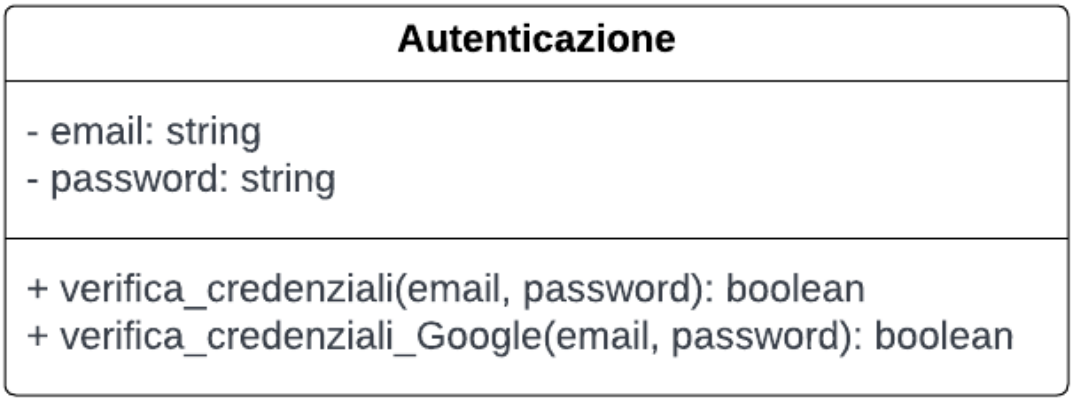

Figura 2. Classe per gestione autenticazione

2.3. Metodo di pagamento

Il diagramma di contesto analizzato presenta un sistema subordinato denominato “Stripe” che si occupa di gestire le richieste di pagamento che vengono effettuate dagli utenti che vogliono inserire l’annuncio “in vetrina”. La classe che si occuperà di tale compito è indicata come “**Pagamento**”, la quale valuterà se approvare o rifiutare le transazioni.

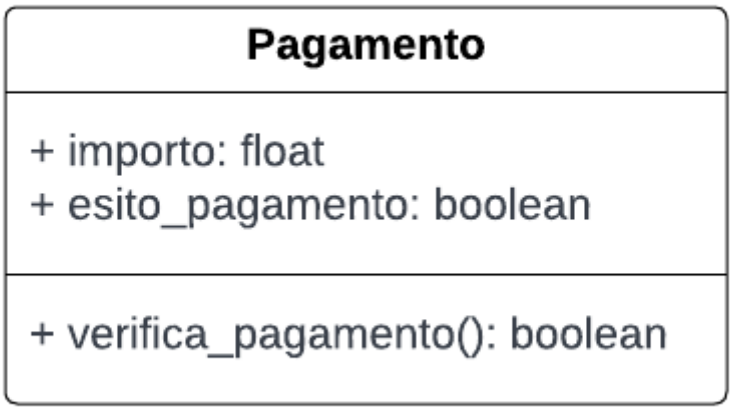

Figura 3. Classe per gestione pagamento

2.4. Gestore Email

Il diagramma di contesto analizzato presenta un sistema subordinato denominato “**Gestore Email**” che si occupa di inviare email all’utente. La classe che descrive questo sistema è “**Gestore Email**”.
Esso si interfaccia con la classe “**Email**”, che rappresenta una email da inviare.

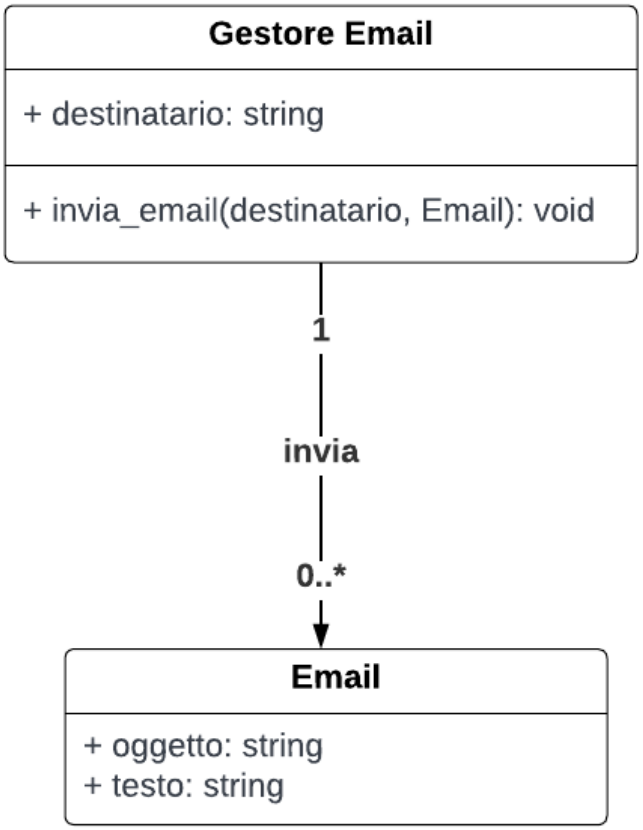

Figura 4. Classe per gestore email e email

2.5. Sistema di messaggistica

Il componente “**Sistema di messaggistica**” gestisce la comunicazione tra utenti attraverso una chat.

La classe “**Chat**” rappresenta una chat tra due “Utenti autenticati". Ogni “Utente autenticato” può interagire con la chat, inviando dei messaggi testuali, descritti dalla classe “**Messaggi**”.
Un “Utente Autenticato” può visualizzare le chat che ha già aperto con altri utenti, oppure iniziarne una nuova con il locatore di un annuncio.

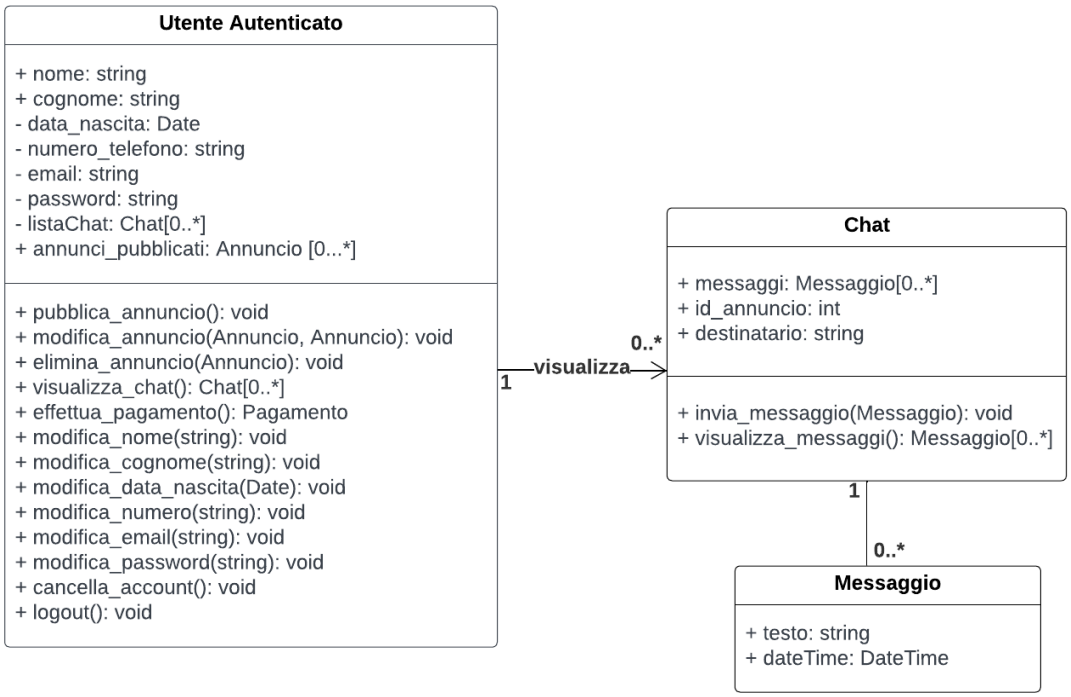

Figura 5. Classi per Chat e Messaggio e la loro interazione con Utente autenticato

2.6. Gestione annunci

Il componente “**Gestione annunci**” descrive come un “Utente generico” interagisce con un “Annuncio”.

La classe “**Annuncio**” descrive tutte le componenti di un annuncio e la classe “**Locale**” descrive le caratteristiche di ogni locale presente nell’immobile.

Dalla pagina di un annuncio, un "Utente Autenticato” può iniziare una chat con il locatore di un annuncio.

Un “Utente Generico” può, visualizzare, cercare e salvare un annuncio, mentre un “Utente Autenticato” può anche pubblicare un annuncio.

Un “Utente Generico” può consultare gli annunci che ha salvato.
Un “Utente Autenticato” ha anche la possibilità di modificare annunci che ha già pubblicato.

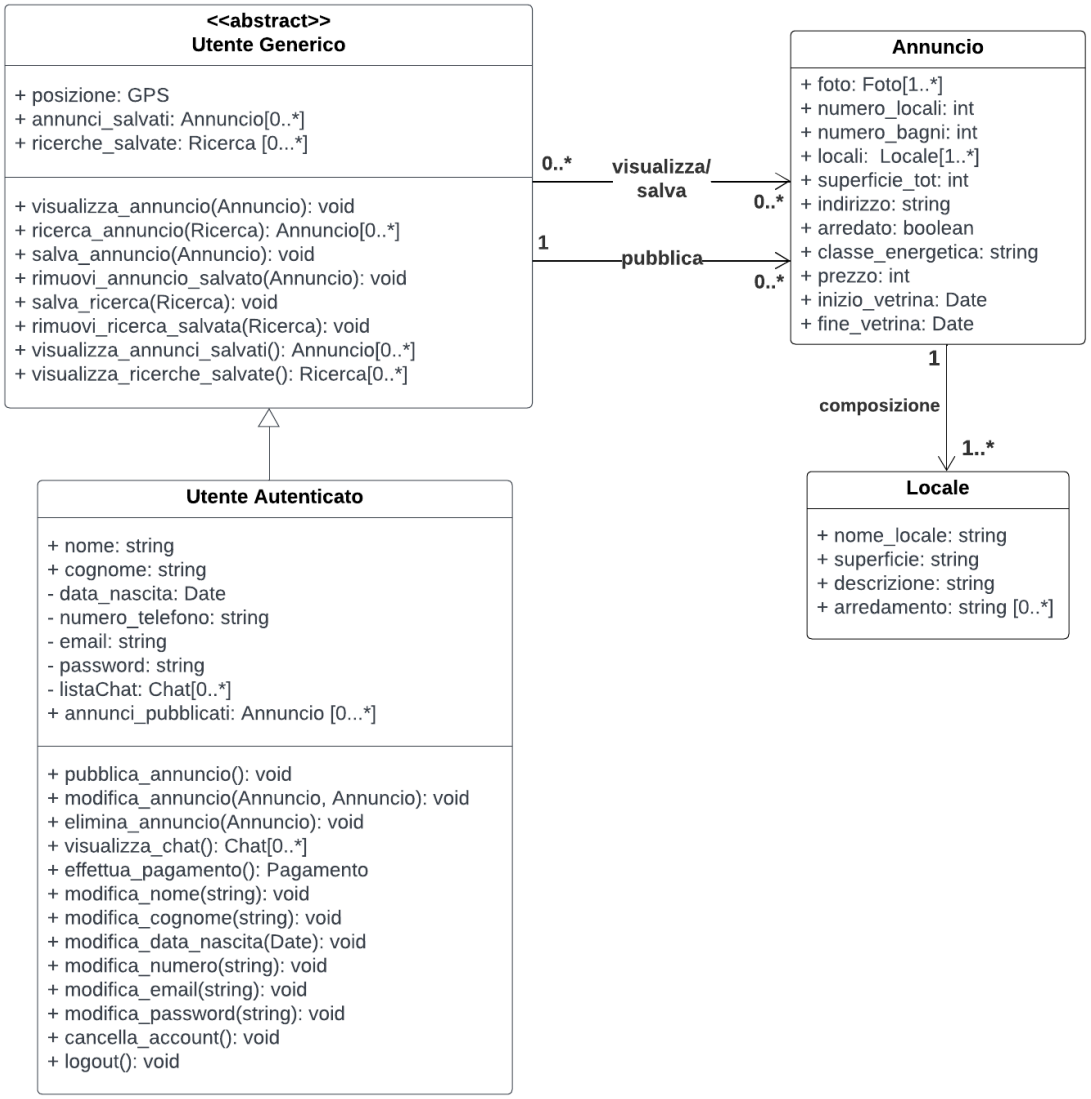

Figura 6. Classi per Annuncio e Locale e la loro interazione con Utente generico

2.7. Home page

Il componente “**Home page**” descrive la pagina iniziale del sito.
Da qui un “Utente Generico” può registrarsi, eseguire il login, effettuare una ricerca per degli annunci, visualizzare un annuncio, salvare un annuncio.
Di default, verranno mostrati annunci in base alla posizione dell’utente oppure, se l’utente non dà il consenso ad utilizzare la sua posizione, gli annunci in ordine di pubblicazione nel sito.

La classe “**Ricerca**” descrive com'è strutturata una ricerca. Essa è composta da filtri descritti con la classe “**Filtro**”.
Quando viene eseguita una “Ricerca”, verranno mostrati solo gli annunci che rispettano i criteri di ricerca.

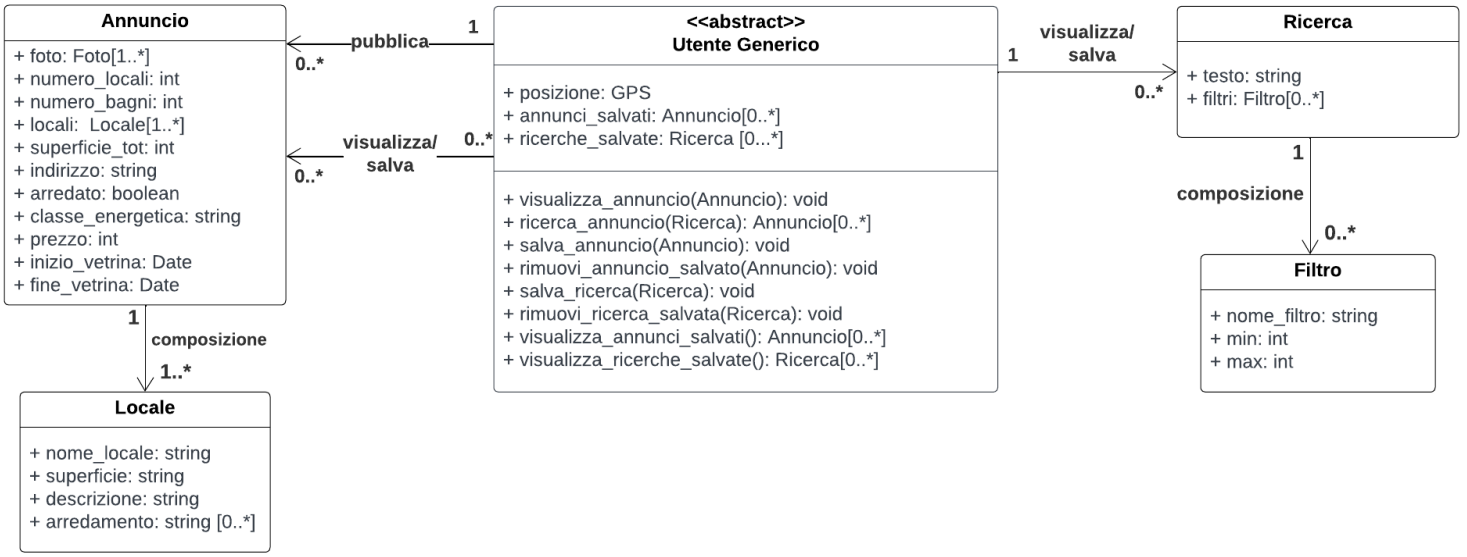

Figura 7. Classe per Ricerca la sua interazione con Utente generico e Annuncio

2.8. Pagina di registrazione/login

I componenti “**Pagina di registrazione**” e “**Pagina di login**” descrivono le pagine in cui un “Utente anonimo” può effettuare la registrazione o il login al sistema.

Queste azioni sono descritte nei metodi della classe “Utente anonimo”.
La correttezza delle credenziali viene verificata dalla classe “Autenticazione” descritta al punto 2.2.

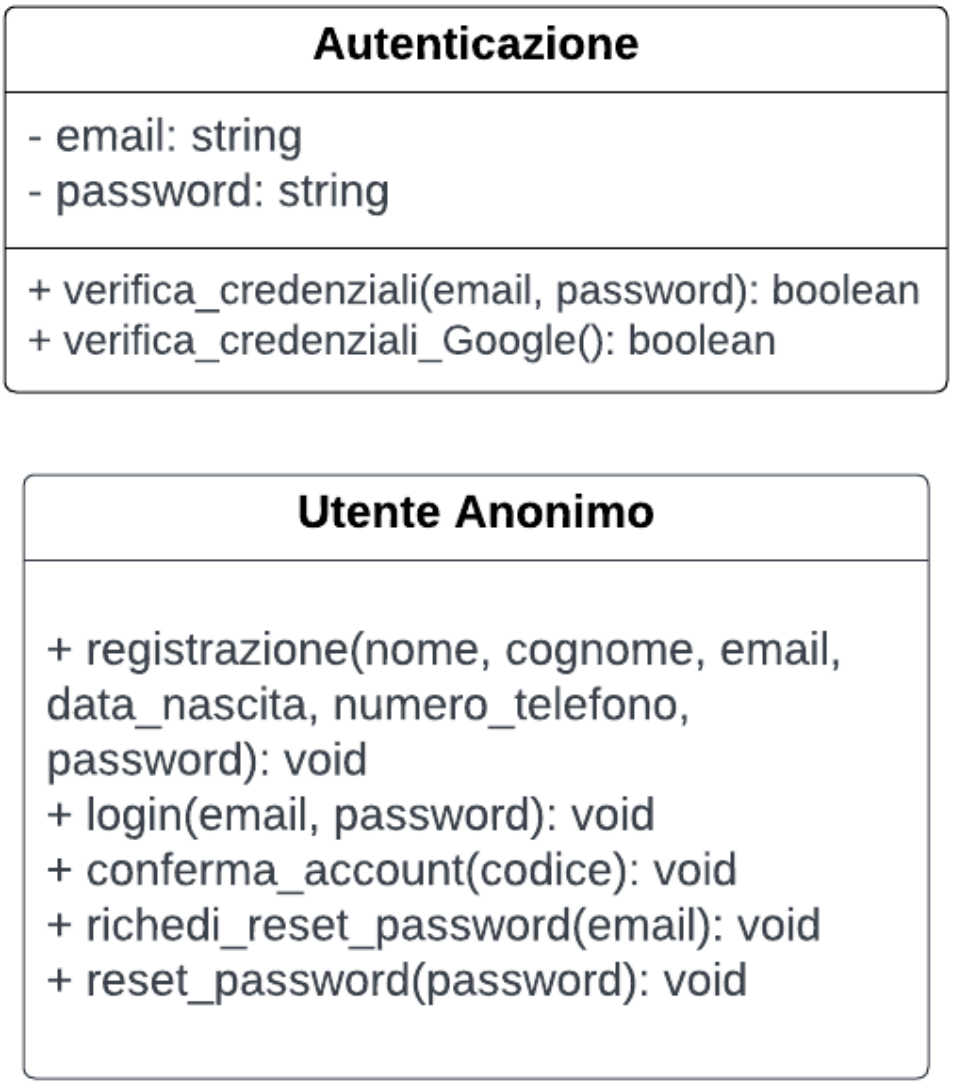

Figura 8. Classi Utente anonimo e Autenticazione

2.9 Diagramma delle classi complessivo

Riportiamo qua di seguito il diagramma delle classi con tutte le classi presenti nel sistema, con i propri metodi, attributi e con tutte le interazioni tra le classi.
Sono presenti anche classi che rappresentano tipi strutturati di attributi nelle classi: **GPS**, **Date** e **Foto**.
GPS è una classe che descrive una posizione geografica, **Date** rappresenta una data particolare, per esempio per la data di nascita, mentre **Foto** invece descrive un file di tipo immagine.

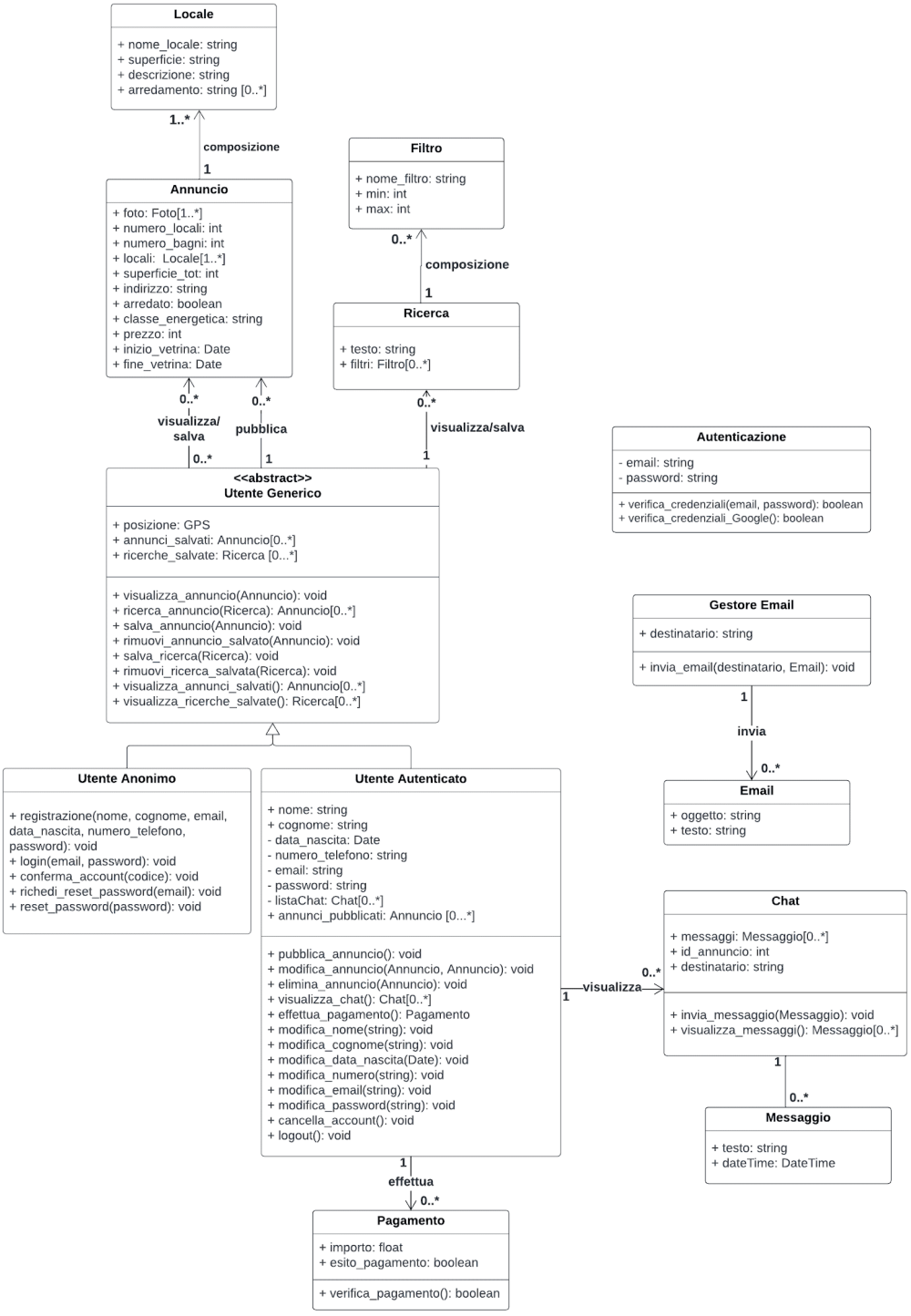

Figura 9. Diagramma delle classi complessivo

3. Codice in Object Constraint Language

In questo capitolo viene descritta in modo formale la logica prevista per alcune operazioni delle classi utilizzando Object Constraint Language (OCL), poiché tali concetti non possono essere espressi in modo formale nel contesto di UML.

3.1 Utente Generico

In **Utente Generico**, il salvataggio di un annuncio o una ricerca, fa sì che l'annuncio o la ricerca venga salvata nel rispettivo attributo che rappresenta la collezione.

```
context UtenteGenerico::salva_annuncio(annuncio)
post:          self.annunci_salvati→includes(annuncio)
context UtenteGenerico::salva_ricerca(ricerca)
post:          self.ricerche_salvate→includes(ricerca)
```

3.2 Utente Autenticato

In **Utente Autenticato**, la pubblicazione di un annuncio aggiunge quest'ultimo nella collezione *annunci_pubblicati*.

Alla fine della modifica dell'annuncio, l'annuncio originale dovrebbe essere sostituito dall' "annuncio nuovo". Esso non è altro che l'annuncio da modificare, con i valori cambiati.

L'eliminazione di un annuncio pubblicato comporta la sua rimozione dagli *annunci_pubblicati*.

```
context UtenteAutenticato::pubblica_annuncio(annuncio)
post:          self.annunci_pubblicati→includes(annuncio)
context UtenteAutenticato::modifica_annuncio
(annuncio_og, annuncio_mod)
post:   self.annunci_pubblicati→includes(annuncio_og) = false
        self.annunci_pubblicati→includes(annuncio_mod) = true
context UtenteAutenticato::elimina_annuncio(annuncio)
post:          self.annunci_pubblicati→includes(annuncio) =
false
```

La modifica dei dati anagrafici o dei dati sensibili comporta la sostituzione di quelli precedenti.

```
context UtenteAutenticato::modifica_nome(nuovo_nome)
post:           self.nome = nuovo_nome
context UtenteAutenticato::modifica_cognome(nuovo_cognome)
post:           self.cognome = nuovo_cognome
context UtenteAutenticato::modifica_data_nascita(nuova_data)
post:           self.data_nascita = nuova_data
context UtenteAutenticato::modifica_numero(nuovo_numero)
post:           self.numero_telefono = nuovo_numero
context UtenteAutenticato::modifica_email(nuova_email)
post:           self.email= nuova_email
context UtenteAutenticato::modifica_password(nuova_password)
post:           self.password = nuova_password
```

3.3 Chat

Nella **Chat**, quando viene inviato un messaggio, esso viene inserito nella collezione *messaggi_inviati*.

```
context Chat::invia_messaggio(messaggio)
post:           self.messaggi_inviati→includes(messaggio)
```

3.4 Pagamento

Un Pagamento di *importo* negativo o nullo non è valido

```
context Pagamento inv: self.importo > 0
```

3.5 Annuncio

Per quanto riguarda la classe **Annuncio**, l'attributo *numero_locali* deve essere coerente con la dimensione di *locali*.

```
context Annuncio inv: self.numero_locali = self.locali→size()
```

Inoltre, *numero_bagni* non può superare *numero_locali*.

```
context Annuncio inv:
self.numero_bagni ⩽ self.numero_locali→size()
```

La *superficie_tot* deve essere almeno uguale quanto la somma delle superfici dei locali.

```
context Annuncio inv:
self.superficie_tot ⩾
self.locali→collect(locale.superficie)→sum()
```

Il *prezzo* deve essere maggiore di 0.

```
context Annuncio inv:
self.prezzo > 0
```

Infine Annuncio non può esistere se non ha almeno una *foto* e almeno un *locale*.

```
context Annuncio inv: foto→size() ⩾ 1
context Annuncio inv: locale→size() ⩾ 1
```

3.6 Locale

Un Locale non può avere una *superficie* nulla o negativa.

```
context Locale inv: self.superficie > 0
```

3.7 Filtro

Un Filtro non può avere come criterio minimo (*min*) un valore negativo.

```
context Filtro inv: self.min ⩾ 0
```

Un Filtro non può avere come criterio massimo (*max*) un valore negativo.

```
context Filtro inv: self.max ⩾ 0
```

Infine un Filtro deve avere il criterio minimo (*min*) minore o uguale al criterio massimo (*max*).

```
context Filtro inv: self.min ⩽ self.max
```

4. Diagramma delle classi con codice OCL

Riportiamo infine il diagramma delle classi con il codice OCL definito.

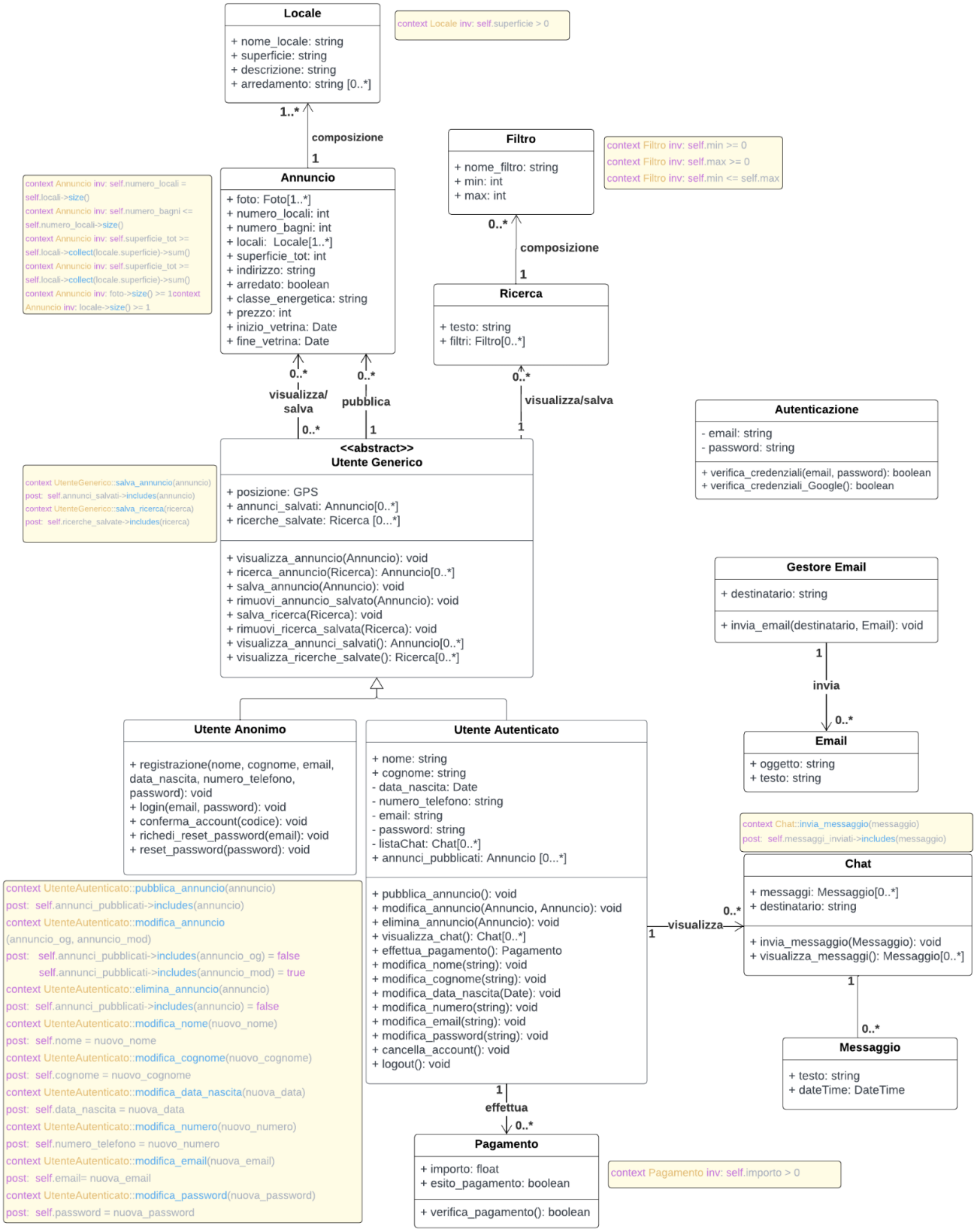

Figura 9. Diagramma delle classi complessivo con codice OCL