UNIVERSITÀ DI TRENTO

Dipartimento di Ingegneria
e Scienza dell'Informazione

Ingegneria del software

Documento di architettura

Mountain Wonders

Gruppo G24

Anno accademico 2023/2024

Indice

1 Scopo del documento

Il presente documento riporta i class diagram del sistema MountainWonders. Lo scopo di questo documento è quello di rappresentare le classi con relative funzionalità che verranno fornite dal sito web. Verranno utilizzati:

- elencare le classi
- fornire una rappresentazione grafica dell'interazione tra le classi

2 Elenco delle classi

In questa sezione mostreremo e documenteremo il diagramma delle classi. Affronteremo una descrizione approfondita dei metodi e dei parametri forniti e fondamentali al fine di realizzare un sito web consono a quello che è stato presentato nei precedenti documenti.

2.1 Database

Procediamo alla definizione della classe per la connessione ad database, che conterrà solamente la stringa di connessione per accedere al database.

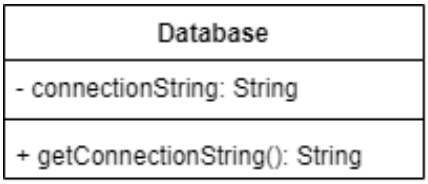

Figura 1: Class diagram database

2.2 Pagine principali

Procediamo alla definizione delle classi che rappresenteranno le pagine principali dell'applicazione web.
La prima è la PaginaUtente che conterrà le info di ciascun utente registrato e i metodi per modificare alcuni di essi.
Segue poi la pagina per la visualizzazione delle montagne e quella per la visualizzazione dei rifugi, che verranno descritte in seguito.

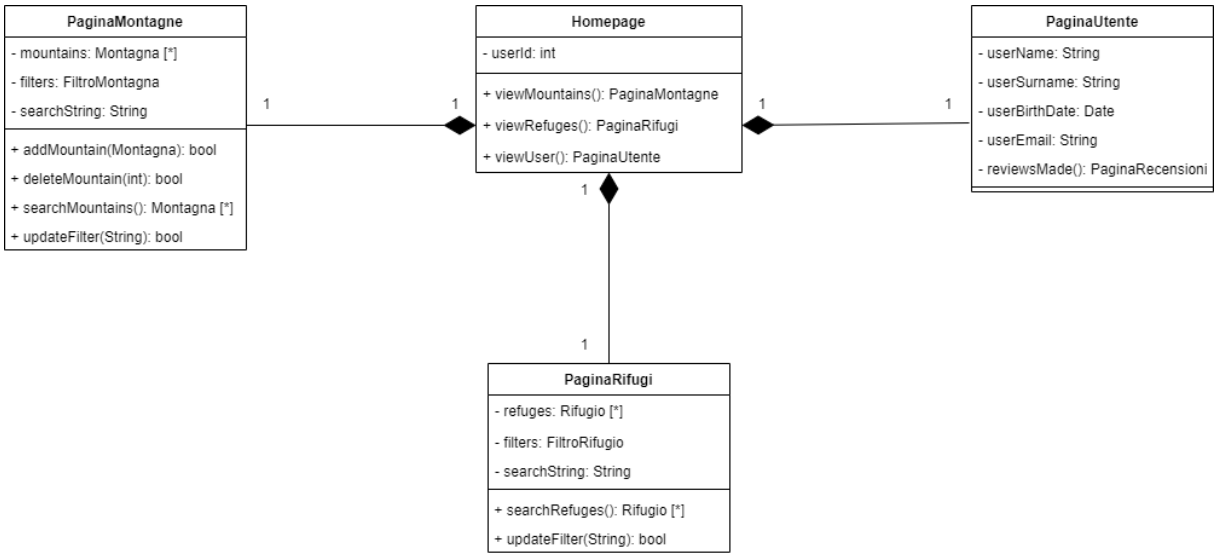

Figura 2: Class diagram pagine principali

2.3 Accounts

Questo insieme di classi racchiude le varie tipologie di account presenti nella pagina web. Esistono infatti 3 tipologie di account:

- AccountAnonimo
- AccountRegistrato
- AccountAmministratore

Le classi AccountAmministratore e AccountRegistrato ereditano gli argomenti della classe AccountAnonimo, non sarà possibile ereditare i metodi register() e login() poiché sono esclusivi della classe AccountAnonimo.
Una volta effettuato il login, in base alla tipologia di utente saranno presenti metodi diversi per ciascun utente.

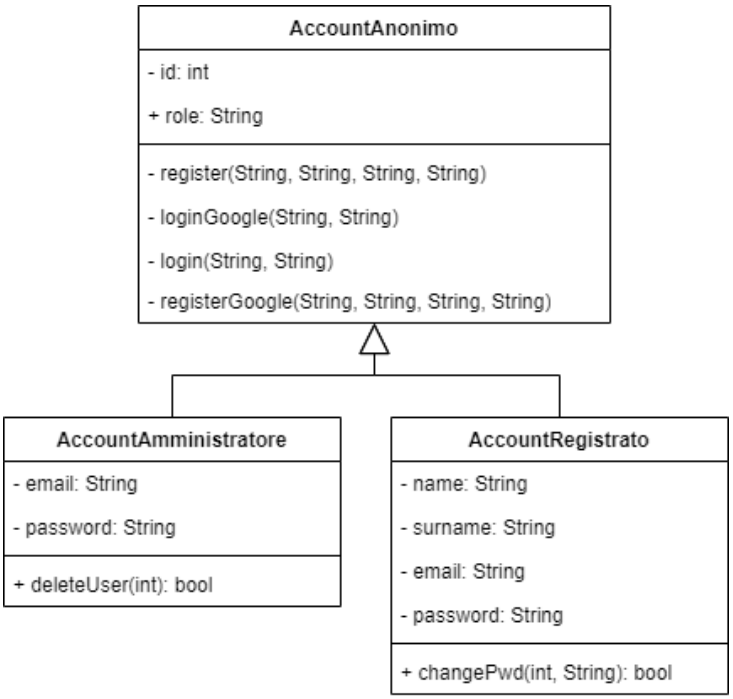

Figura 3: Class diagram accounts

2.4 Montagna

Procediamo alla definizione delle classi relative alle montagne.
La classe PaginaMontagne andrà a contenere l'elenco delle montagne presenti all'interno del database, e i relativi filtri applicati al momento della ricerca in modo da mostrare i dati in base alle preferenze dell'utente.
La pagina Montagna, invece, rappresenta l'oggetto del mondo reale che verrà salvato all'interno del database, dotato di tutti gli attributi necessari.

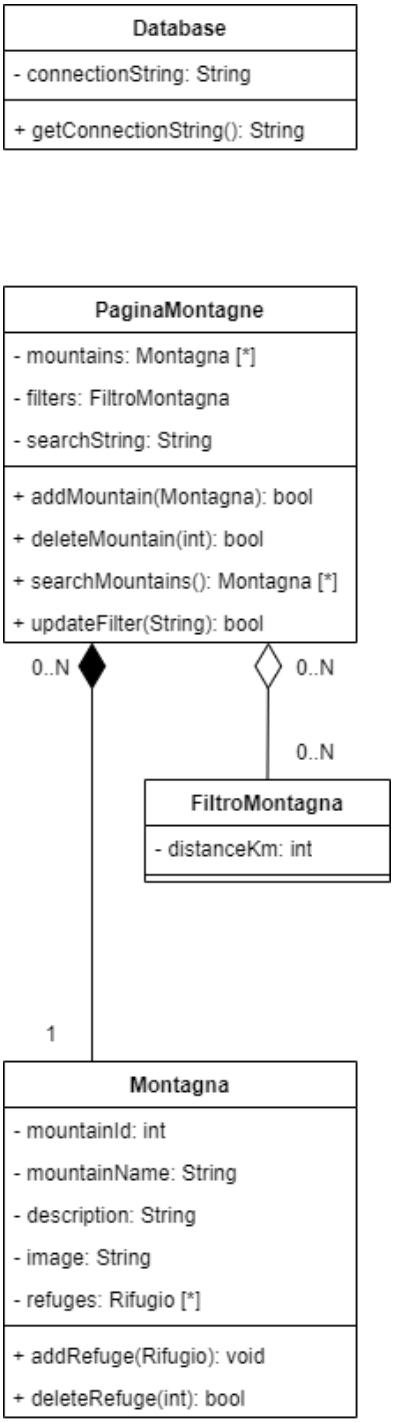

Figura 4: Class diagram montagne

2.5 Rifugio

Le due classi presentate in seguito contengono le pagine per le funzionalità dei rifugi.
PaginaRifugi è la classe che rappresenta la pagina web con l'elenco di tutti i rifugi (scremati se vengono applicati dei filtri oppure tutti i rifugi presenti nel database nel caso contrario).
Rifugio è la classe che contiene le informazioni di ogni singolo rifugio insieme all'insieme di recensioni che sono state effettuate per ognuno di essi.

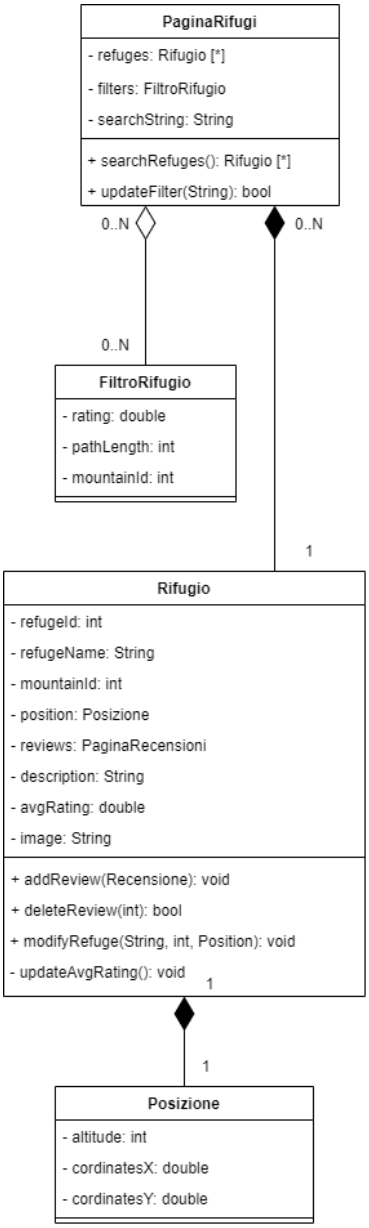

Figura 5: Class diagram rifugi

2.6 Recensione

All'interno del seguente insieme vengono rappresentate le classi necessarie per gestire le recensioni.
La classe PaginaRecensioni contiene e mostrerà tutte le recensioni effettuate da un utente.
La classe Recensione contiene tutte le informazioni relative ad una recensione.

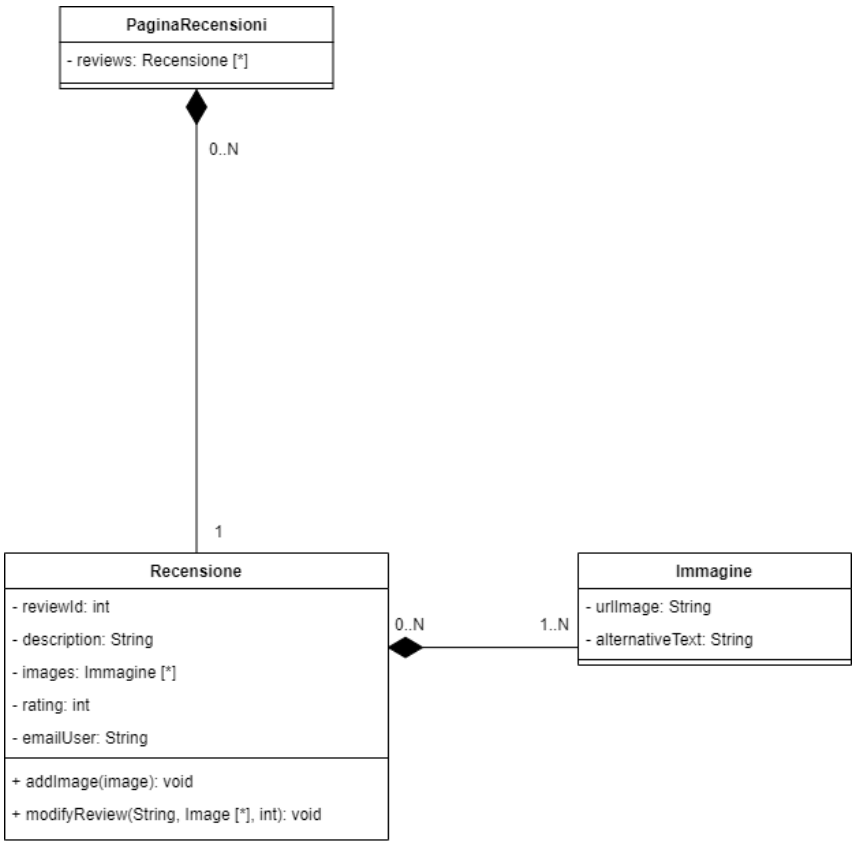

Figura 6: Class diagram recensioni

2.7 Filtri

All'interno del seguente insieme vengono rappresentate le classi necessarie per gestire i filtri.
La classe FiltroMontagna contiene i filtri applicabili ad una montagne, mentre la classe FiltroRifugio contiene i filtri applicabili ad un rifugio.
Tra i filtri dei rifugi c'è anche la montagna a cui appartiene, in modo da poter mostrare tutti i rifugi di una determinata montagna.

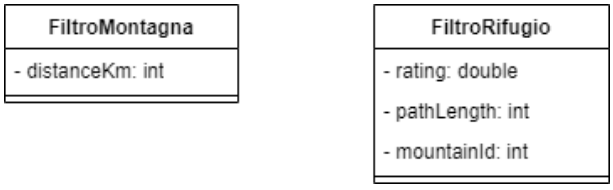

Figura 7: Class diagram filtri

2.8 Class diagram totale

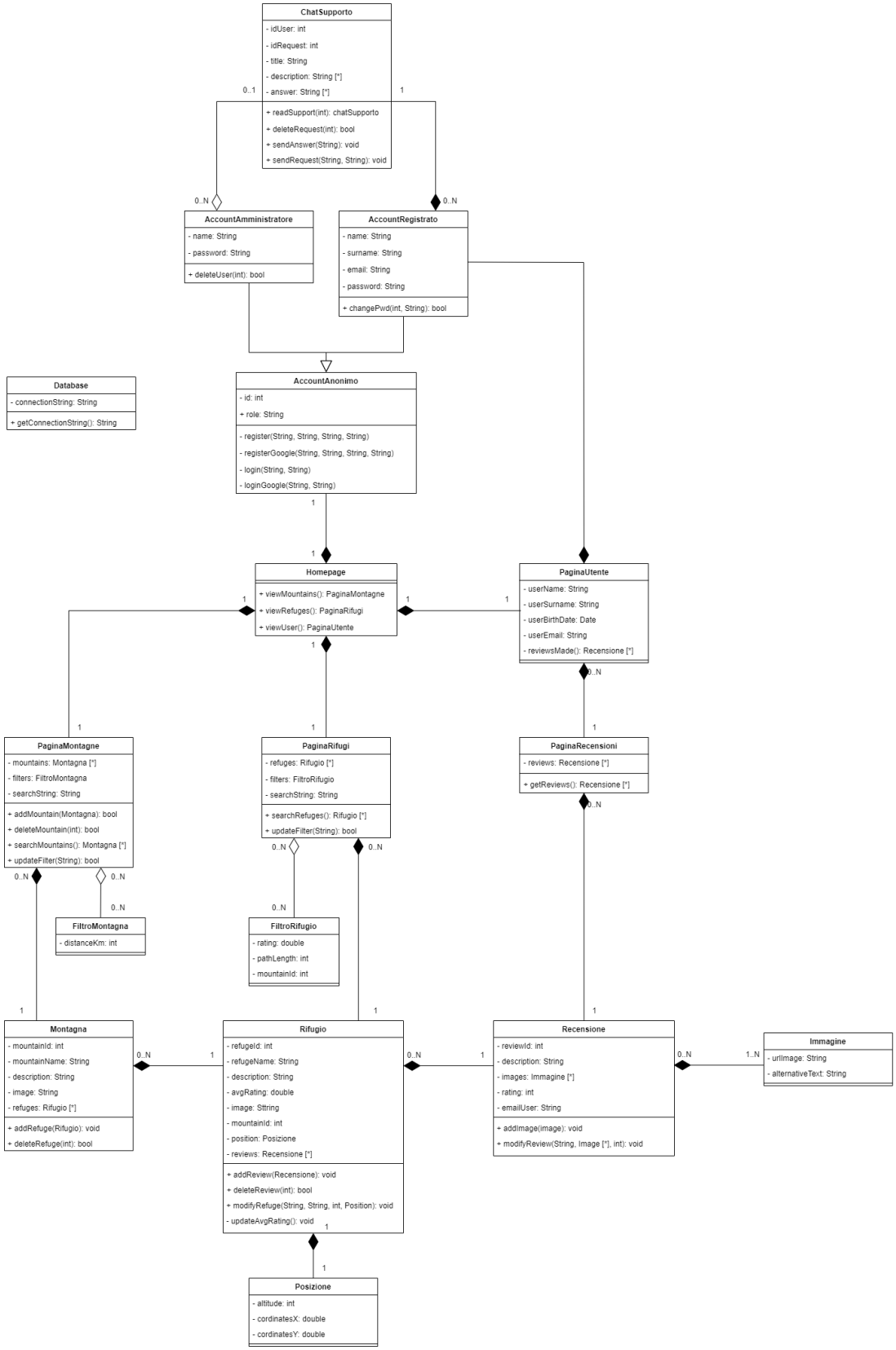

Figura 8: Class diagram totale

3 OCL

In questa sezione verranno rappresentate le interazioni tra le classi utilizzando OCL (Object Contraint Language) in modo da poter descrivere concetti non rappresentabili tramite diagramma delle classi.

3.1 PaginaMontagne

- L'addMountain() e la deleteMountain() di una montagna può essere eseguito solo se l'accesso è stato fatto dall'amministratore, la deleteMontagna() è possibile solo se sono presenti montagne e corrispondono alla montagna da eliminare.

  In linguaggio OCL:
  context PaginaMontagna::addMountain(Montagna: montagna)
  pre: AccountAnonimo.role == "Admin"
  post: PaginaMontagna.mountains = PaginaMontagna.mountains+montagna

  context PaginaMontagna::deleteMontagna(int: idMontagna)
  pre: PaginaMontagna.mountains != NULL and PaginaMontagna.mountains.id == idMontagna and AccountAnonimo.role == "Admin"
  post: PaginaMontagna.mountains.id = NULL

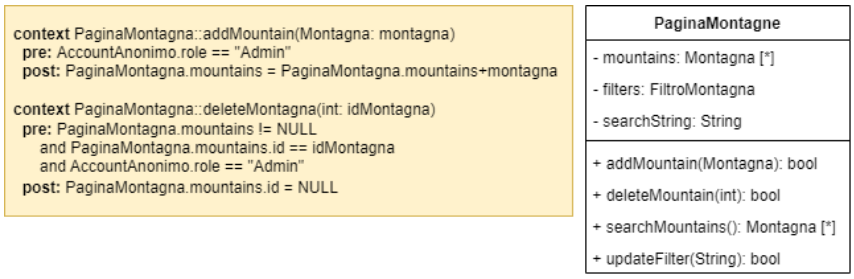

Figura 9: OCL Pagina Montagne

3.2 Montagna

- addRefuge() è una funzionalità che può essere eseguita solo da un account registrato.

  In linguaggio OCL:
  context Montagna::addRefuge(Rifugio: rifugio)
  pre: AccountAnonimo.role == "Registrato"
  post: Montagna.refuges= Montagna.refuges+rifugio

- deleteRefuge() ha un parametro int che indica l'id del rifugio che deve essere eliminato, per questo motivo deve essere maggiore di 0, questa funzionalità è esclusiva di un ammministratore. Quando si elimina un rifugio vengono eliminate tutte le recensioni legate al rifugio.

In linguaggio OCL:
context Montagna::deleteRefuge(int: idRifugio)
pre: AccountAnonimo.role == "Admin"
and Montagna.refuges.refugeId == idRifugio
post: Montagna.refuges.reviews = NULL and Montagna.refuges.refugeId = NULL

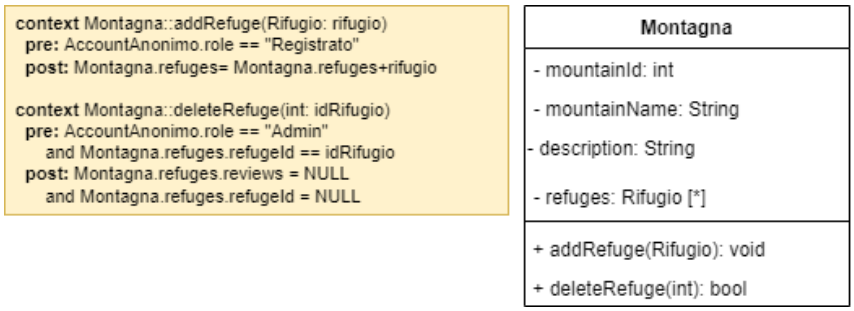

Figura 10: OCL Montagna

3.3 Rifugio

- addReview() è una funzionalità che può essere eseguita solo da un account registrato. Una volta aggiunta la recensione verrà chiamato il metodo updateAvgRating() in modo tale da aggiornare il valore precedente di avgRating.

  In linguaggio OCL:
  context Rifugio::addReview(Recensione: recensione)
  pre: AccountAnonimo.role == "Registrato"
  post: Rifugio.reviews = Rifugio.reviews+recensione and Rifugio::updateAvgRating()

- deleteReview() è una funzionalità che può essere eseguita solo da un account registrato o amministratore e solo se esiste una recensione a cui si fa riferimento.

  In linguaggio OCL:
  context Rifugio::deleteReview(int: idRecensione)
  pre: AccountAnonimo.role == "Registrato"
  and Recensione.reviews.reviewId==idRecensione
  post: Rifugio.reviews.reviewId = NULL

- modifyRefuge() è una funzionalità esclusiva dell'amministratore, ha tre parametri che contengono i nuovi valori da assegnare agli attributi che possono essere modificati.

  In linguaggio OCL:
  context Rifugio::modifyRefuge(String: nomeRifugio, int idMontagna, Position: posizione)
  pre: AccountAnonimo.role == "Admin"
  post: Rifugio.refugeName = nomeRifugio and Rifugio.mountainId = idMontagna and Rifugio.position=posizione

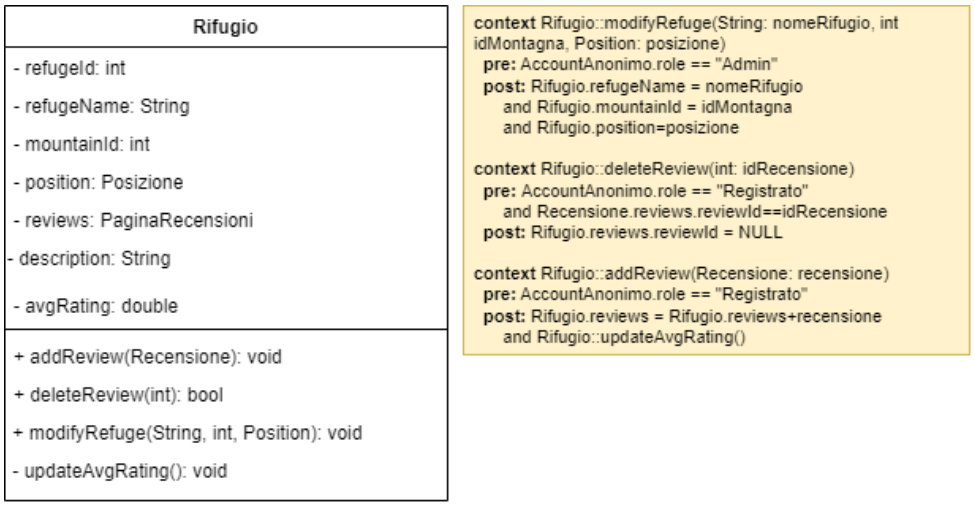

Figura 11: OCL Rifugio

3.4 Recensione

- Rating recensione compreso da 1 a 5.

  In linguaggio OCL:
  context Recensione **inv**: Recensione.rating >= 1 and Recensione.rating <= 5

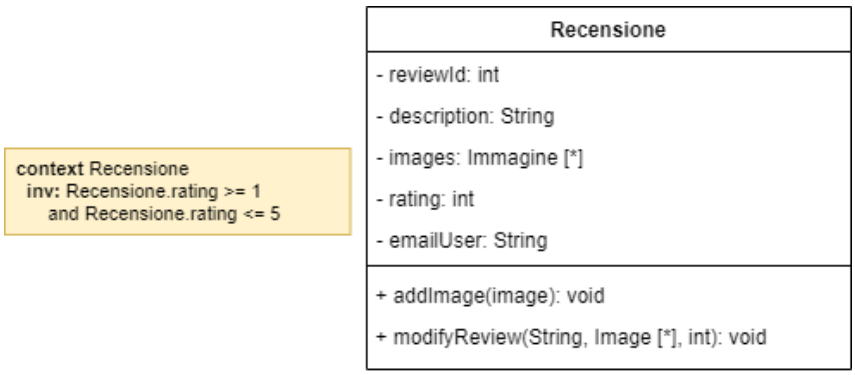

Figura 12: OCL Recensione

3.5 Posizione

- Le coordinate vengono ottenute grazie al servizio di google maps da cui è possibile selezionare la posizione di un luogo.
- Altitudine del rifugio compresa da 0 a 5000 metri.

  In linguaggio OCL:
  context Posizione **inv**: Posizione.altitude >= 0 and Posizione.altitude <= 5000

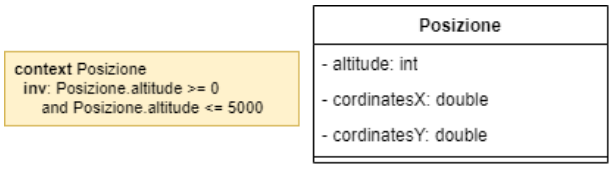

Figura 13: OCL Posizione

3.6 Homepage

- Nel caso in cui l'utente non abbia effettuato il login, la funzione viewUser() dell'homepage reindirizzerà l'utente nella pagina di login.

  In linguaggio OCL:
  context AccountAnonimo.role
  pre: AccountAnonimo.role == "Anonimo"
  post: AccountAnonimo::login()
  pre: AccountAnonimo.role != "Anonimo"
  post: Homepage::viewUser()

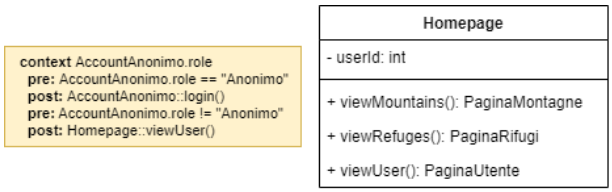

Figura 14: OCL Homepage

3.7 ChatSupporto

- Non è possibile richiedere supporto (sendRequest()) se non si è un utente registrato

  In linguaggio OCL:
  context ChatSupporto.sendRequest() **inv** AccountAnonimo.role == "Registrato"

- L'idUser è un numero maggiore di zero ed è relativo ad un account registrato.

  In linguaggio OCL:
  context ChatSupporto **inv** ChatSupporto.idUser >= 1

- L'idRequest è un numero maggiore di zero relativo al numero identificativo della richiesta fatta dall'utente

  In linguaggio OCL:
  context ChatSupporto **inv** ChatSupporto.idRequest >= 1

- description e answer sono degli array perché è possibile avere uno scambio di messaggi di supporto

- Una sendAnswer() può essere fatta solo da un account amministratore

  In linguaggio OCL:
  context ChatSupporto.sendAnswer() **inv** AccountAnonimo.role == "Admin"

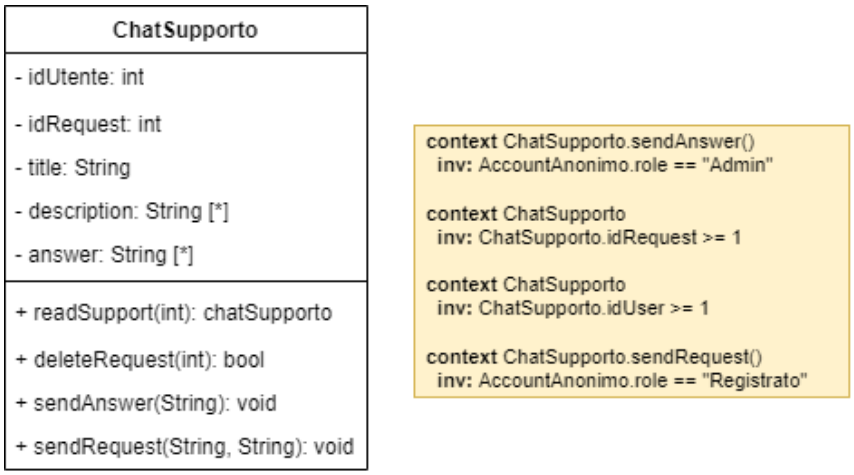

Figura 15: OCL Chat supporto

3.8 AccountAnonimo

- Il login può essere effettuato solo da un utente che ha già eseguito la registrazione in precedenza e quindi le sue credenziali sono già presenti all'interno del database.

  In linguaggio OCL:
  context AccountAnonimo.login()
  inv AccountAnonimo.role == "Anonimo"

- Le funzioni registerGoogle, loginGoogle vengono effettuate utilizzando le API google, i controlli vengono fatti da servizi esterni (Google).

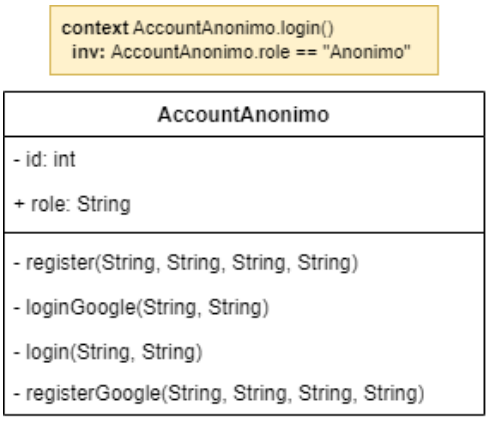

Figura 16: OCL Account anonimo

3.9 Diagramma delle classi con OCL

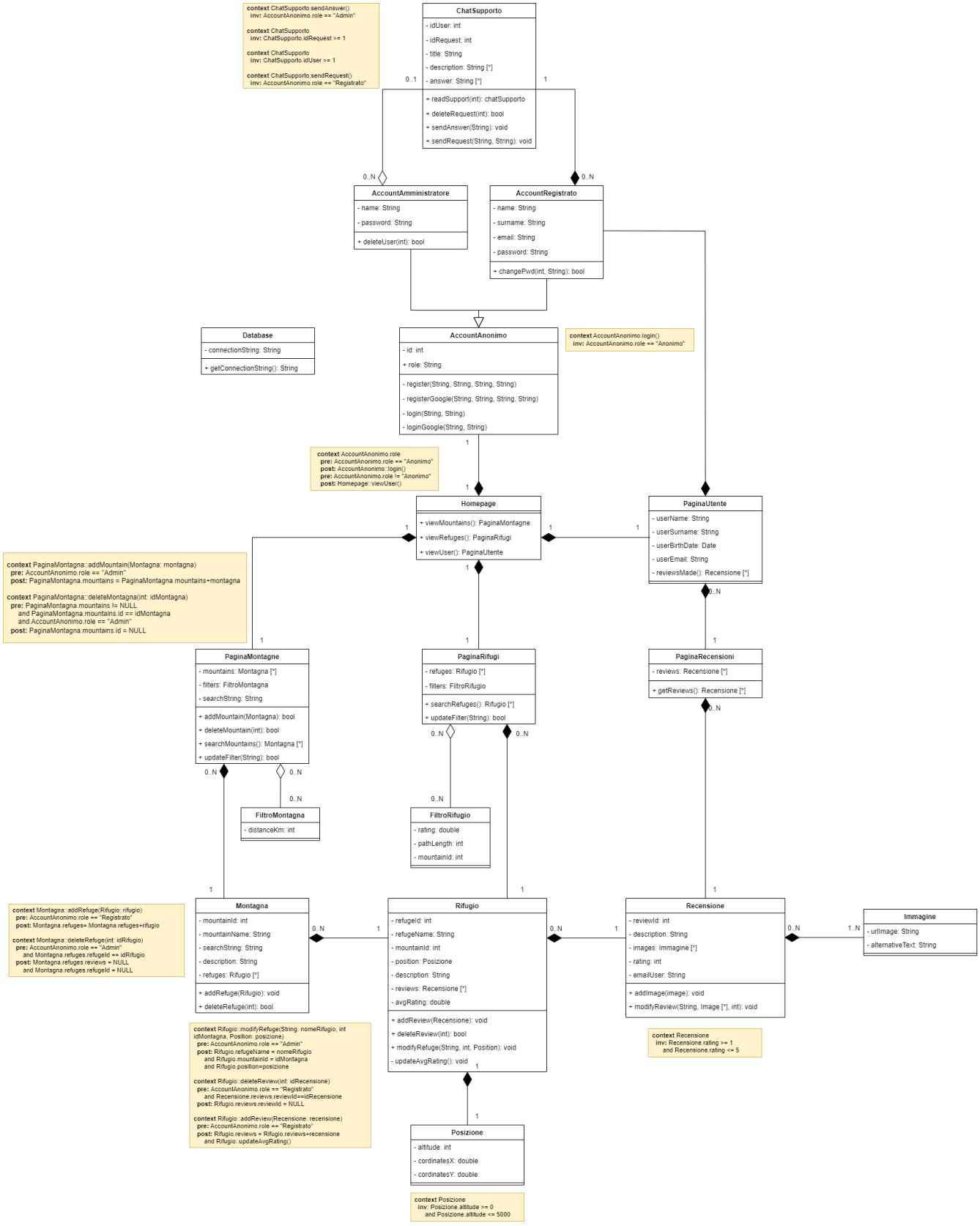

Figura 17: Class diagram con OCL