Skupply

UNIVERSITY OF TRENTO
Dipartimento di Ingegneria e Scienza dell'Informazione

Documento di Architettura

Skupply

Versione 1.0
01 Dicembre 2022

Di Zepp Dorijan
dorijan.dizepp@studenti.unitn.it

Piccin Andrea
andrea.piccin@studenti.unitn.it

Rossi Simone
simone.rossi-2@studenti.unitn.it

D3

Indice

1. Scopo del documento

Il presente documento riporta la definizione dell'architettura del progetto Skupply, in particolare il diagramma delle classi espresso in linguaggio UML e una sua implementazione logica grazie al codice OCL.
Nel documento precedente sono stati presentati i diagrammi degli Use Case, il diagramma di contesto e il diagramma dei componenti, partendo da questi viene definito il diagramma delle classi con la funzione di dettagliare sia le modalità implementative del codice sia il suo funzionamento logico.

2. Diagramma delle classi

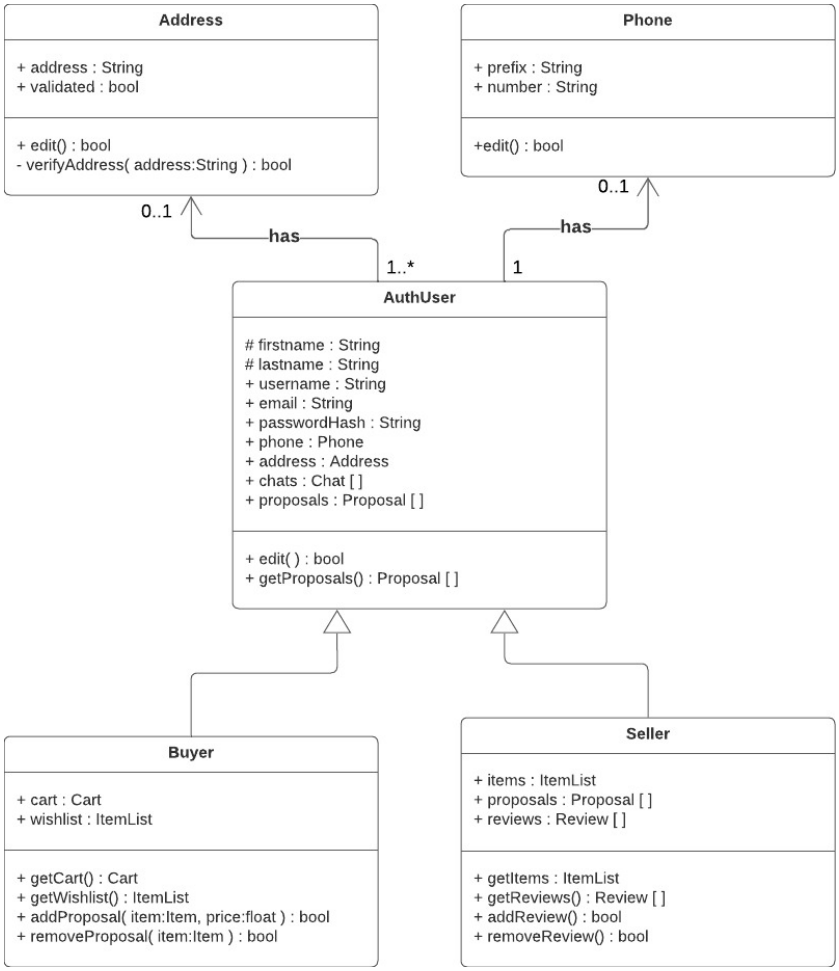
Vengono ora riportate le classi previste dal progetto Skupply.
Ogni componente del precedente documento è suddiviso in una o più classi in modo da ottenere un'architettura con una complessità bilanciata, una corretta separazione delle responsabilità e di facile gestione e manutenzione.
Ogni classe è identificata da un nome, alcuni attributi ed eventuali metodi; ogni classe ha relazioni con altre classi in base a servizi forniti e/o utilizzati e/o relazioni di ereditarietà.

2.1. Utenti

Analizzando il diagramma di contesto si evidenziano la due entità:

- acquirente ("Buyer"): è il normale utilizzatore dell'applicazione che svolge l'azione di acquistare un prodotto.
- venditore ("Seller"): è la tipologia di utente che permetta di pubblicare nuovi articoli sul sito e di gestirne vendita e spedizione.

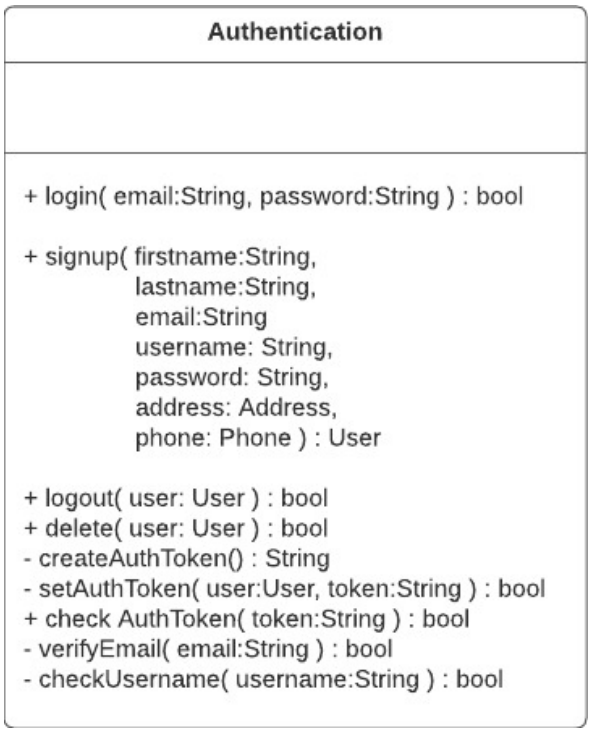
Queste due classi riportano molti attributi e metodi comuni per i quali si è deciso di implementare una nuova classe "AuthUser" generalizzazione di "Buyer" e "Seller".



Vengono introdotte anche la classe per la gestione degli indirizzi "Address" e la classe "Phone" per i contatti telefonici.
La classe "Address" fornisce un metodo per validare l'indirizzo tramite API terze.

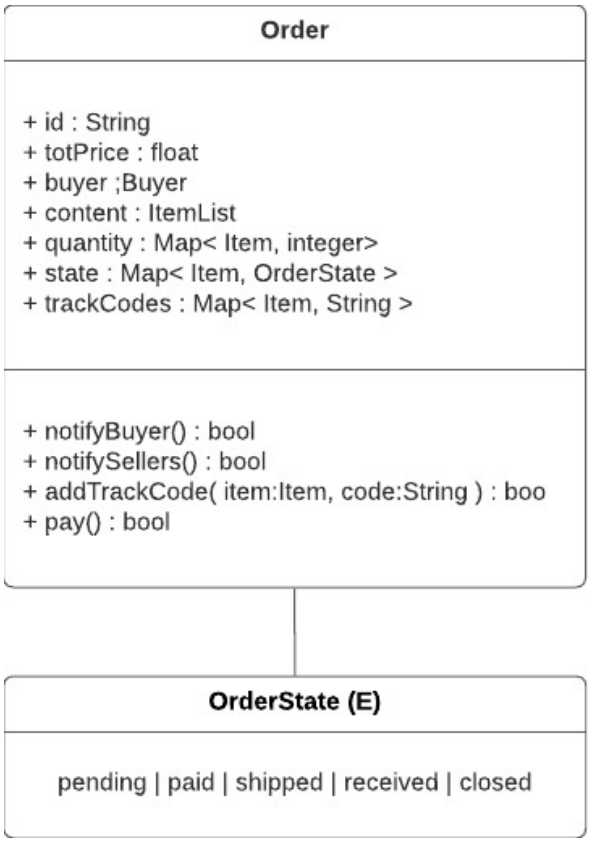
2.2. Autenticazione

La classe "Authentication" si occupa delle procedure di login, registrazione, logout e cancellazione del profilo come da diagramma dei componenti.
Questa classe si occuperà della gestione dei token che permettono ad un utente di identificarsi e di stabilire la validità della sua connessione.



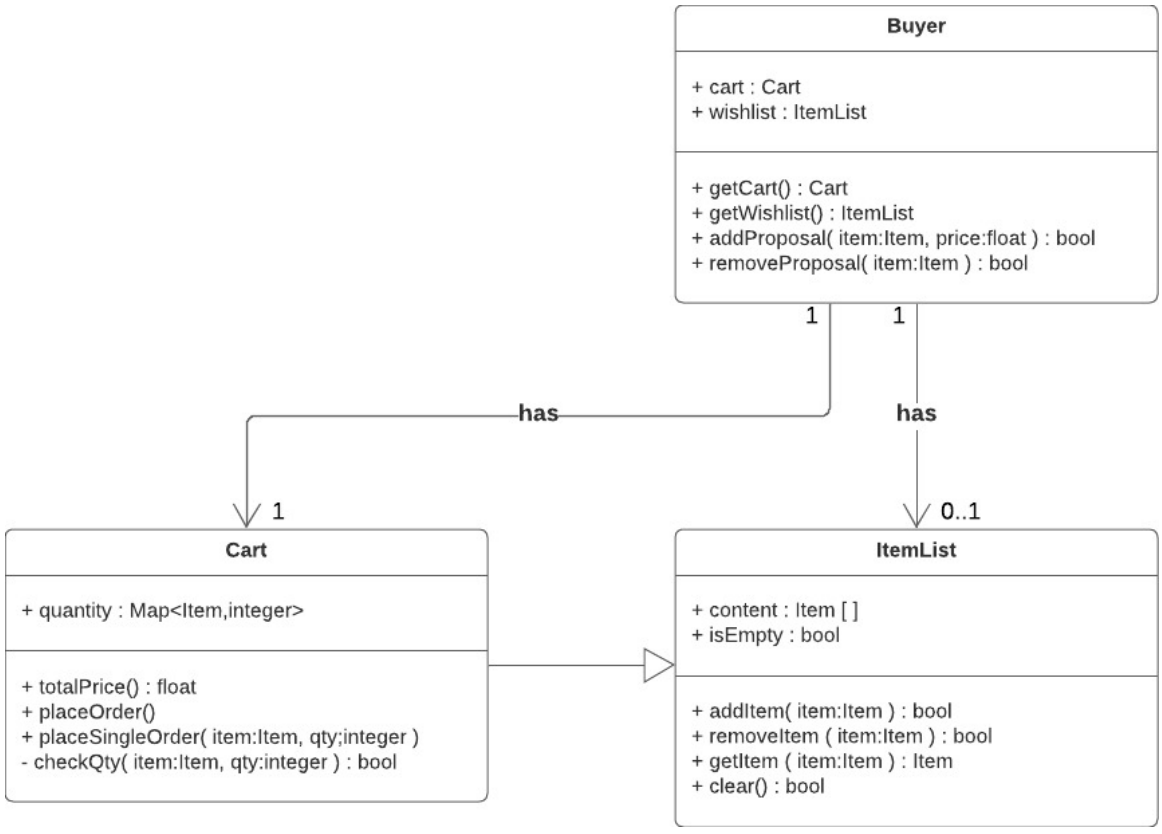
2.3. Ordini e Pagamenti

Nel diagramma di contesto e nel diagramma dei componenti vengono individuati più entità che permettono la gestione dei pagamenti (tramite servizi esterni) e degli ordini, da qui viene individuata la classe "Order".
Questa classe si occupa di gestire il checkout di un insieme di prodotti e quindi di modificarne le quantità disponibili, notificare i venditori della vendita e della necessità di comunicare il codice di spedizione, notificare l'acquirente e tracciare le spedizioni. Si interfaccia con API esterne per la gestione del pagamento.



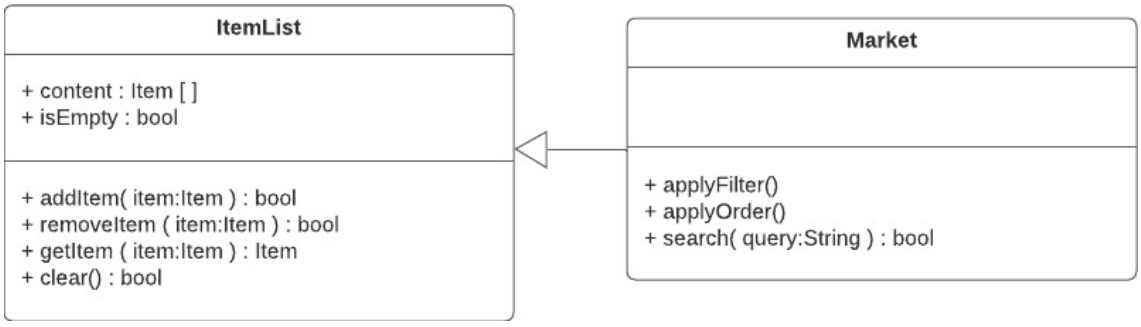
2.4. Carrello e Wishlist

Dal diagramma dei componenti vengono individuate le classe carrello ("Cart") e il componente wishlist i quali permettono la gestione di una lista di contenuti, in questo caso, articoli ("Items").
In quanto i componenti presentano molte similitudini si è creata la classe "ItemList" generalizzazione di carrello. La wishlist sarà implementata come istanza di "ItemList".



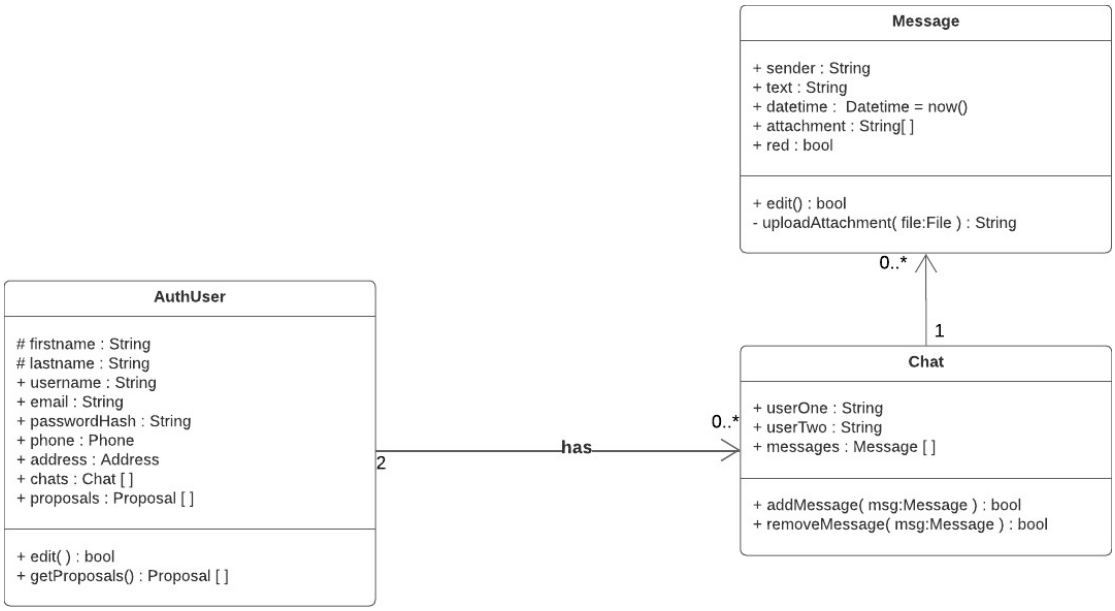
2.5. Negozio

Dal diagramma dei componenti si evidenzia la necessità di una classe negozio ("Market") la quale gestisce la visualizzazione degli annunci in base ai termini di ricerca inseriti dall'utente e degli eventuali filtri e ordinamenti utilizzati.
Come il carrello, anche il negozio, eredita attributi e metodi dalla classe "ItemList".



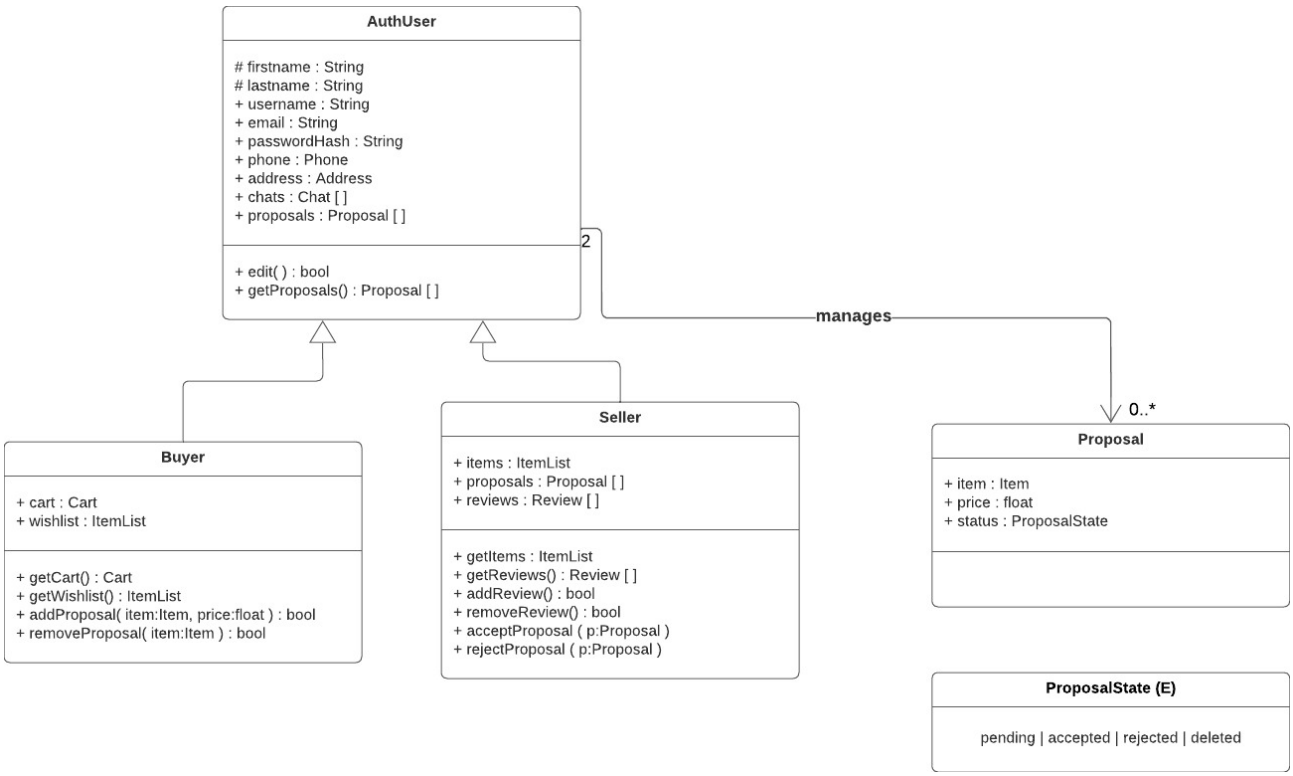
2.6. Chat

Dal diagramma dei componenti e dal diagramma di contesto viene individuata la necessità di dover creare delle classi per poter permettere la possibilità di scambiare messaggi e allegati tra i vari utenti all'interno della relativa chat.
Si vengono a formare due nuove classi: "Message" e "Chat". Ogni utente ("AuthUser") possederà un elenco di chat ognuna delle quali conterrà una serie di messaggi.



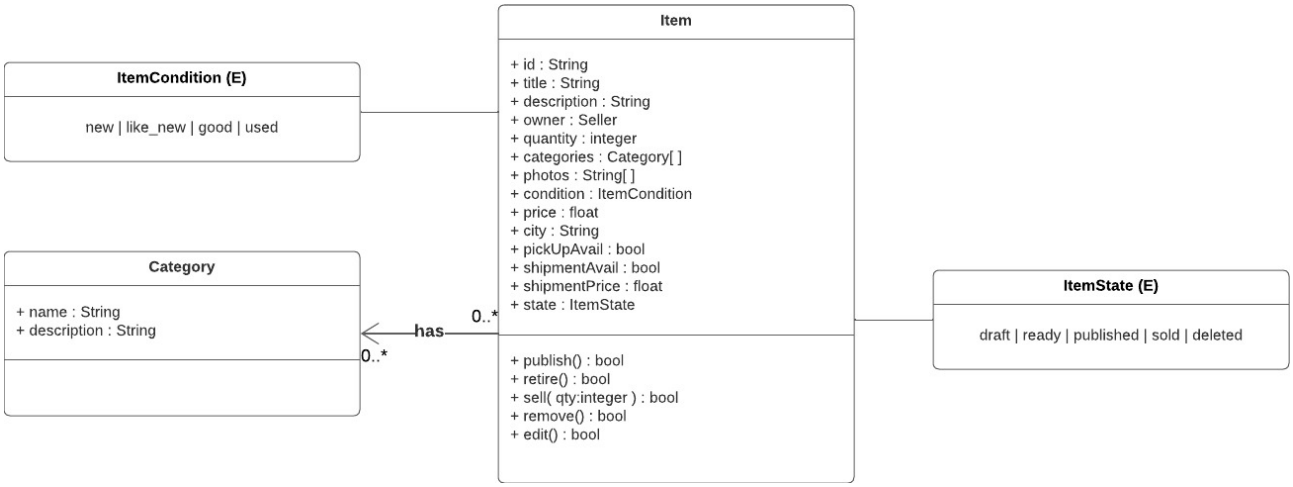
2.7. Proposte

Si nota la necessità di creare una classe che permette la gestione delle proposte di acquisto, ovvero la possibilità (da parte di un acquirente) di creare una offerta da sottoporre al venditore che potrà, in seguito, accettarla o rifiutarla.
Viene creata la classe proposta che avrà relazioni con la classe "AuthUser" (di cui "Buyer" e "Seller" sono figlie) ma dove solo la classe "Buyer" potrà creare nuove proposte e solo la classe "Seller" potrà accettare o rifiutare una proposta.



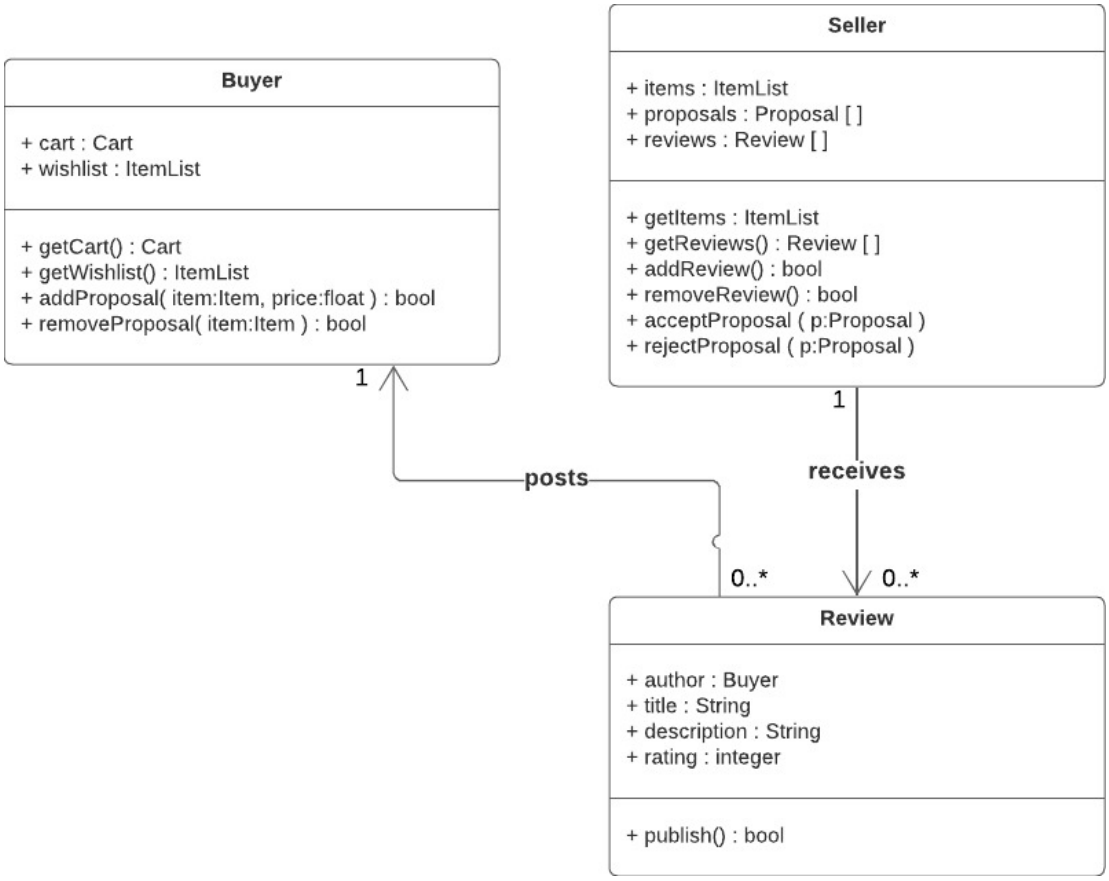
2.8. Annunci

Viene creata la classe Annuncio che rappresenta l'entità più importante del sistema Skupply. Tale classe detiene tutti gli attributi necessari per la rappresentazione più dettagliata dell'annuncio e mette a disposizione le funzionalità denotate sia dalle interfacce del componente Gestore Annunci che sia dal componente Visualizzatore Annuncio.

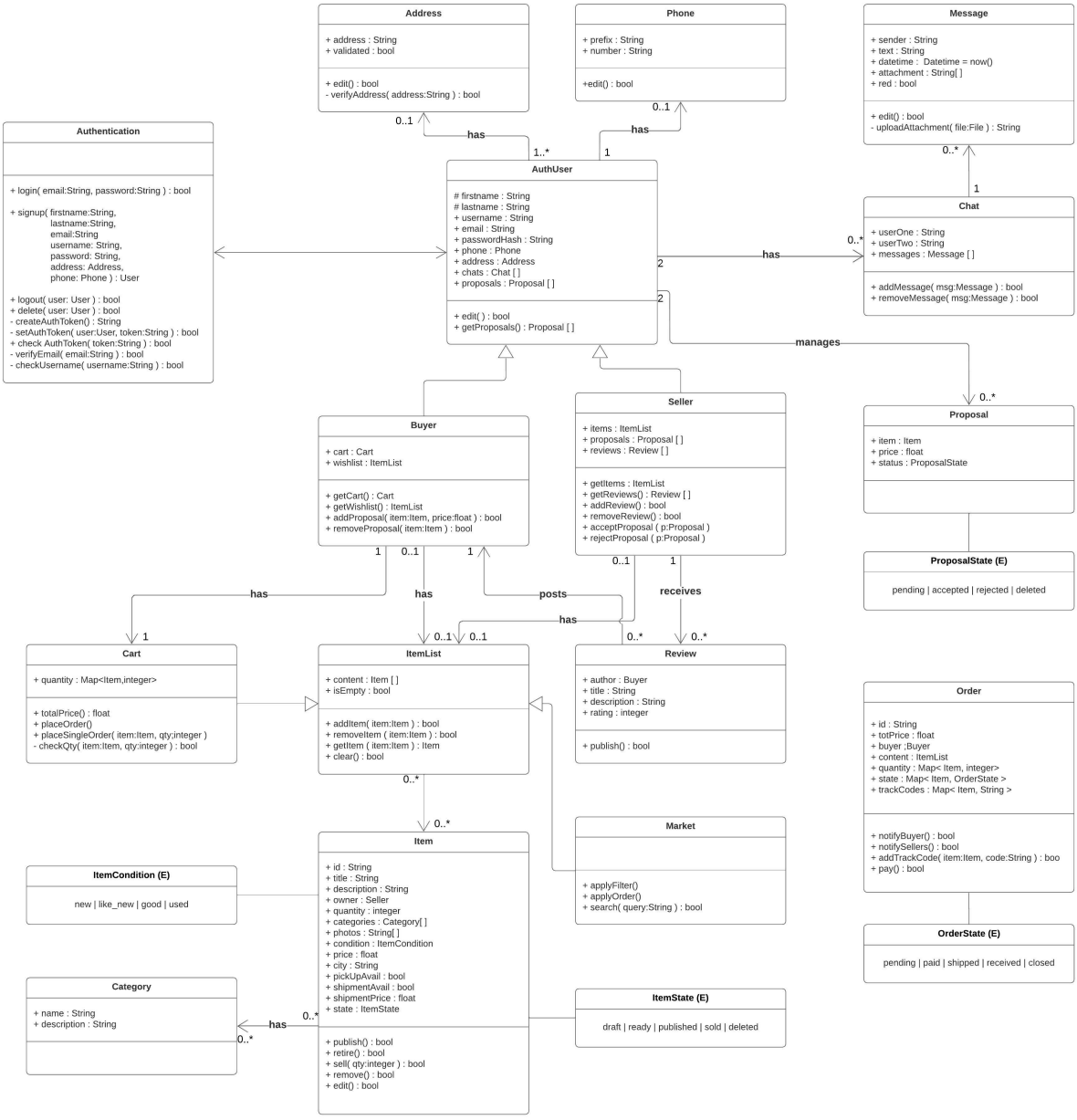


2.9. Recensioni

Vista la possibilità, da parte di un acquirente, di lasciare una recensione sulla sua esperienza con un determinato venditore viene inserita la classe recensione ("Review").



2.10. Diagramma finale



3. Object Constraint Language (OCL)

In questa sezione è descritta in modo formale la logica prevista nell'ambito di alcune operazioni in alcune classi attraverso lo Object Constraint Language (OCL).

3.1. Autenticazione

Di seguito sono riportati i costraints riguardanti questa specifica classe:

a. Nell'ambito della registrazione gli attributi firstname, lastname, email, username e password devono essere non nulli.
   Questo requisito viene espresso dalla seguente espressione in OCL:

```
context Authentication :: signup( )
pre:  firstname != null and
      lastname != null and
      email != null and
      username != null and
```

3.2. Ordini e pagamenti

Di seguito sono riportati i costraints riguardanti questa specifica classe:

a. Nella classe Order non è mai possibile che il prezzo totale sia minore o uguale a zero

```
context Order inv : totPrice > 0.00
```

b. Non è possibile che nessun prodotto sia presente in un ordine

```
context Order inv : quantity.size() > 0
```

3.3. Carrello e wishlist

Di seguito sono riportati i costraints riguardanti questa specifica classe:

a. Nella classe Cart, nel procedere alla creazione di un ordine il numero e/o la quantità dei prodotti dev'essere diversa da zero.

```
context Cart :: placeOrder()
pre: !(quantity.isEmpty or content.isEmpty)
```

```
context Cart :: placeSingleOrder()
pre: qty > 0
```

b. Nella classe ItemList per eliminare un Item è necessario che la lista sia non vuota

```
context ItemList :: removeItem()
pre: !isEmpty
```

c. Nella classe ItemList per l'operazione di svuotamento della lista è necessario che la lista sia non vuota e, invece, dev'esserlo a procedura completata

```
context ItemList :: clear()
pre: !isEmpty
post: isEmpty
```

3.4. Chat

Di seguito sono riportati i costraints riguardanti questa specifica classe:

a. È possibile rimuovere un messaggio da una chat se e solo se l'insieme dei messaggi è non vuoto

```
context Chat :: removeMessage()
pre: messages.size() > 0
```

b. La data di invio di un messaggio non può essere futura

```
context Message inv : date <= now()
```

3.5. Proposte

Di seguito sono riportati i costraints riguardanti questa specifica classe:

a. Il prezzo stabilito per una proposta dev'essere strettamente positivo

```
context Proposal inv : price > 0.00
```

3.6. Annunci

Di seguito sono riportati i costraints riguardanti questa specifica classe:

a. La quantità di un item dev'essere strettamente positiva o uguale a zero sono nel caso l'articolo sia nello stato di venduto

```
context Item inv : quantity > 0 or (quantity==0 and state==SOLD)
```

b. Il prezzo di un articolo dev'essere maggiore di 0

```
context Item inv : price > 0.00
```

c. La pubblicazione di un articolo porta lo stato a pubblicato

```
context Item :: publish()
post: state == PUBLISHED
```

d. Il ritiro di un oggetto dalla vendita riporta l'oggetto allo stato di ready, ovvero pronto per essere pubblicato

```
context Item :: retire()
post: state == READY
```

e. La vendita di un oggetto è possibile se è nello stato di published, se la sia quantità è maggiore o uguale alla quantità richiesta e porta ad un decremento della quantità disponibile pari alla quantità acquistata

```
context Item :: sell( qty:integer )
pre: quantity >= qty and state == PUBLISHED
post: quantity == quantity - qty
```

f. L'operazione di rimozione dell'articolo porta l'item in stato delete

```
context Item :: remove()
post: state == DELETED
```

3.7. Recensioni

Di seguito sono riportati i costraints riguardanti questa specifica classe:

a. La valutazione dev'essere di tipo intero e compresa tra 0 e 5 estremi inclusi

```
context Review inv : 0 <= rating and rating <= 5
```

b. Titolo e Descrizione di una recensione non possono essere vuoti

```
context Review inv : title != null and description != null
```

3.8. Address e Phone

Di seguito sono riportati i costraints riguardanti questa specifica classe:

a. Il numero di telefono deve contenere 10 caratteri, mentre il prefisso 3

```
context Phone inv : number.length == 10 and prefix.length == 3
```

b. La verifica dell'indirizzo può essere effettuata su un indirizzo non ancora validato e l'attributo validate deve sempre essere il risultato della funzione verifyEmail

```
context Address :: verifyEmail()
pre: !verified
```

```
context Address inv : visited == verifyEmail( email )
```

3.9. Diagramma finale