Urban Shirt

Vendita e personalizzazione t-shirt con spazio per la community

15/11/2022

D3

Team 32

Minatel Davide

Hu Angela

Zanoni Devis

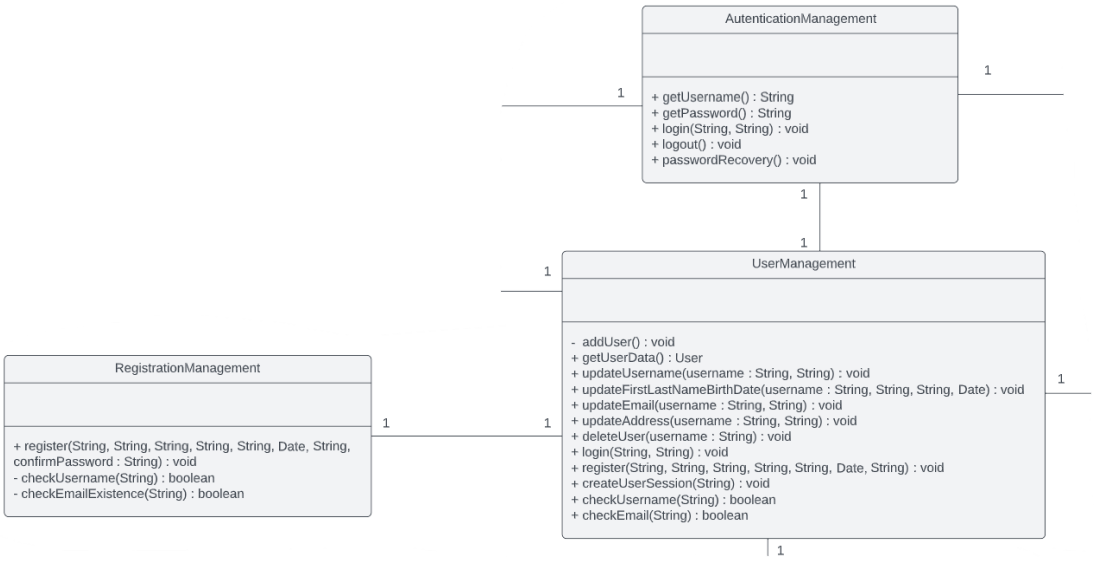
Indice

Diagramma delle Classi

Nei seguenti paragrafi viene presentato il diagramma delle classi, che rappresenta l'architettura del sistema sulla quale si basa l'implementazione.
Di seguito verranno descritte le varie sezioni del diagramma

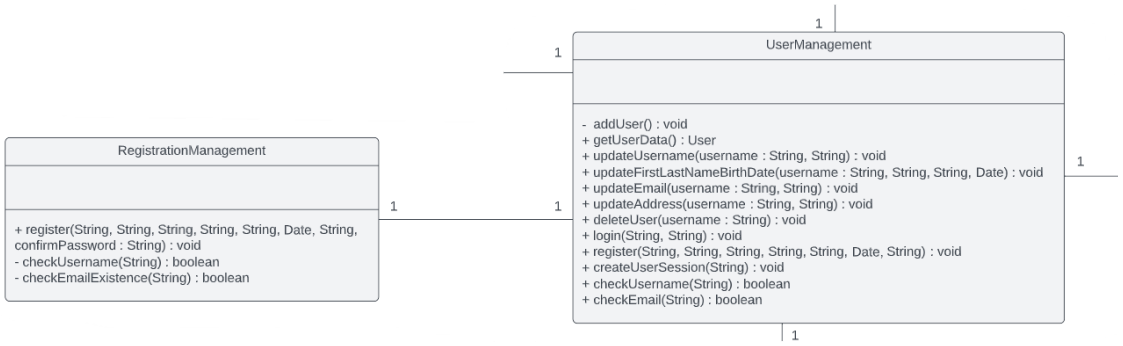
Gestione autenticazione

Dal diagramma dei componenti è stata identificata una classe *AutenticationManagement*. La classe si interfaccia con la sezione di sistema "*Gestione dati utente*", gestita dalla classe *UserManagement*. Questa classe, dedicata all'autenticazione, controlla la correttezza delle credenziali inserite a livello formale, evita che non vengano lasciati campi vuoti o che vengano inseriti caratteri non ammessi. Successivamente comunica le credenziali ricevute alla sezione *Gestione Dati Utente*, la quale controlla la veridicità dei suddetti dati interrogando il database. Se le credenziali sono corrette restituisce alla classe autenticazione un esito positivo, generando un token per la sessione dell'utente e l'autenticazione avviene con successo. In caso contrario la classe invia un messaggio di errore da far visualizzare all'utente.



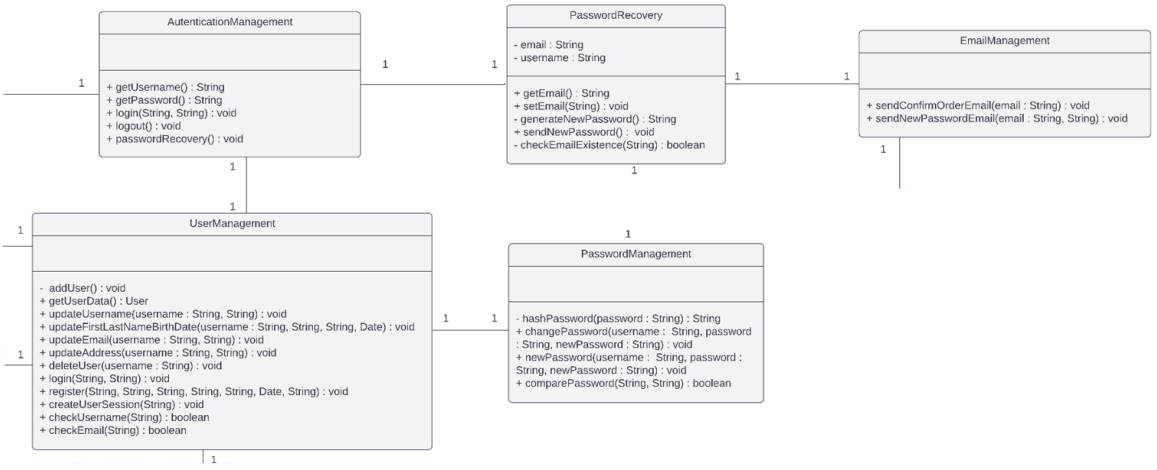
Gestione registrazione

Dal diagramma dei componenti è stata identificata una classe *RegistrationManagement*, che si interfaccia con la sezione di sistema di "*Gestione dati utente*", gestita dalla classe *UserManagement*. *RegistrationManagement* controlla la correttezza dei dati inseriti secondo i vari canoni specificati nel punto **RF 7.1** dell'analisi dei requisiti. Comunica, successivamente, i dati ricevuti alla sezione "*Gestione Dati Utente*", la quale procede con l'inserimento nel database e la generazione di un codice hash associato alla password. Se l'inserimento va a buon fine, viene restituito alla classe di registrazione un esito positivo, rimandando l'utente alla pagina di login. In caso contrario, la classe invierà un messaggio di errore da far visualizzare all'utente.



Recupero password

Dal diagramma dei componenti è stata identificata una classe *PasswordRecovery*, che si interfaccia con le sezioni di sistema "*Gestione dati utente*", rappresentata dalla classe *UserManagement*, e "*Gestione Mail*", individuato con la classe *EmailManagement*. Questa classe dedicata al recupero password viene utilizzata per generare in autonomia una nuova password da mandare poi all'utente che l'ha richiesta. La classe riferisce la nuova password e l'email a *EmailManagement*, che si occuperà di fornire la nuova credenziale di accesso all'utente, e a *UserManagement*, tramite *PasswordManagement,* che aggiornerà i dati presenti sul database.

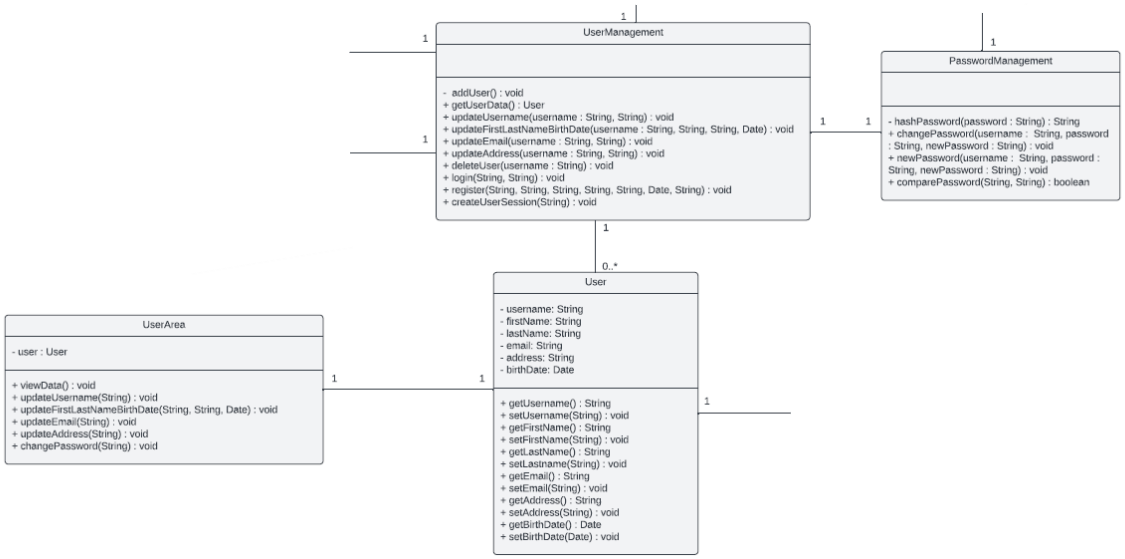


Gestione profilo utente

Dal diagramma dei componenti è stata identificata una classe *UserArea*, che si interfaccia con la sezione del sistema "*Gestione Dati Utente*", e la sezione dedicata alla modifica password *"Interfaccia Modifica Password"*. Tramite questa gestione, l'utente può compiere diverse azioni relative al proprio profilo, quali visualizzarne e modificarne i dati, eliminare il proprio profilo o modificare la password.

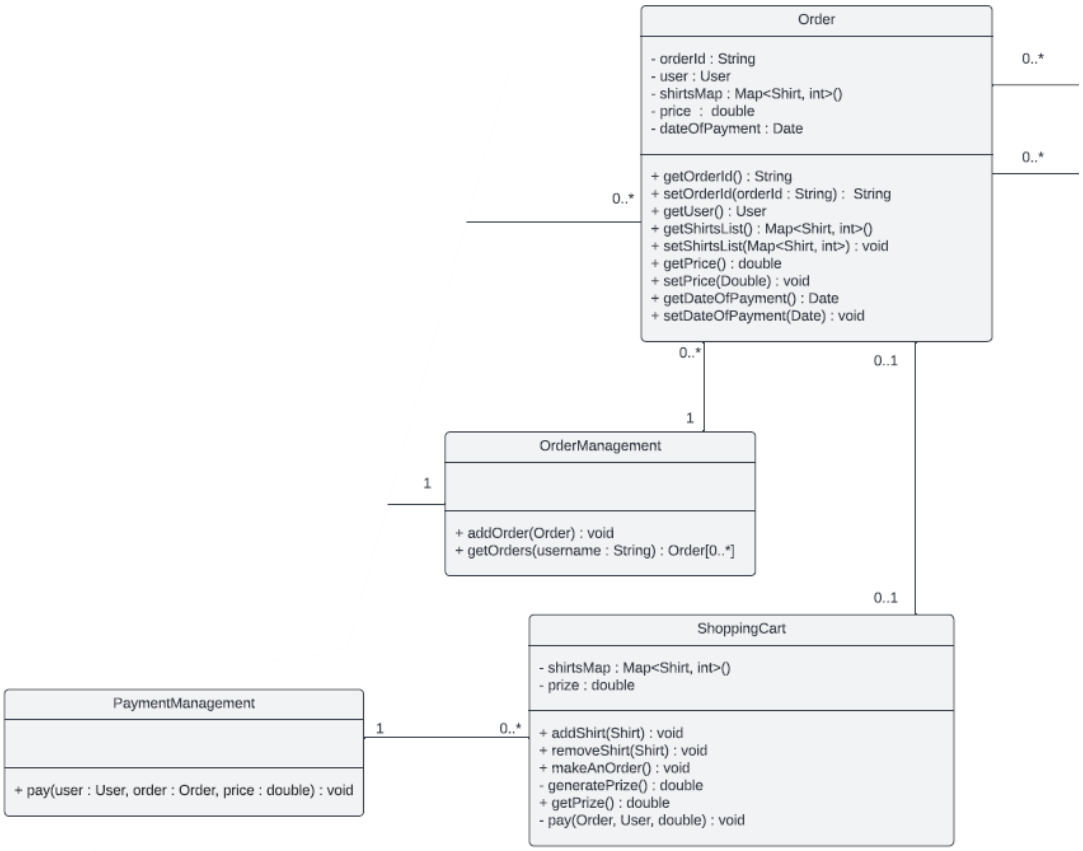
Nel caso in cui si desideri modificare la password, l'utente viene indirizzato verso l'interfaccia dedicata, gestita dalla classe *PasswordManagement*, che gli permetterà di effettuare la modifica di quest'ultima con gli opportuni controlli: la correttezza della

vecchia password, la congruenza tra la nuova password e la conferma, nel rispetto dei requisiti definiti nel punto **RNF 4** dell'analisi dei requisiti.
Per tutte le altre azioni invece la classe comunicherà l'operazione da eseguire alla classe *UserManagement*, che ne restituirà l'esito. Si precisa che per la modifica dei dati dell'utente, la classe invierà i nuovi dati per aggiornare quelli attuali.



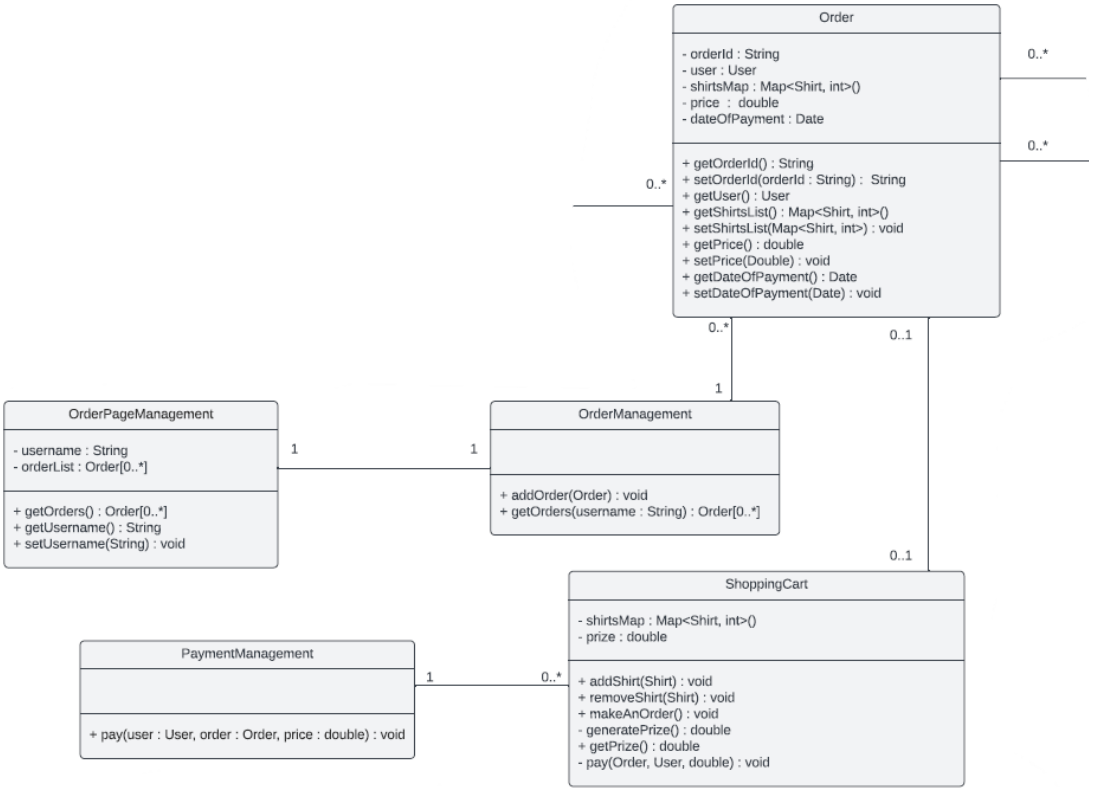
Gestione pagamento

Dal diagramma dei componenti è stata identificata una classe *PaymentManagement*, che si interfaccia con la sezione di sistema "*Gestione Ordini*", rappresentato dalla classe *OrderManagement*.
PaymentManagement si occupa di gestire la transazione effettiva del pagamento tramite il portale di PayPal come descritto nel punto **RF 1** dell'analisi dei requisiti: riceve le informazioni necessarie e comunica con il gateway per procedere con l'operazione.
A pagamento avvenuto restituisce l'esito dell'operazione a *"Gestione Ordini",* passando per le classi intermedie e notifica successivamente l'utente.



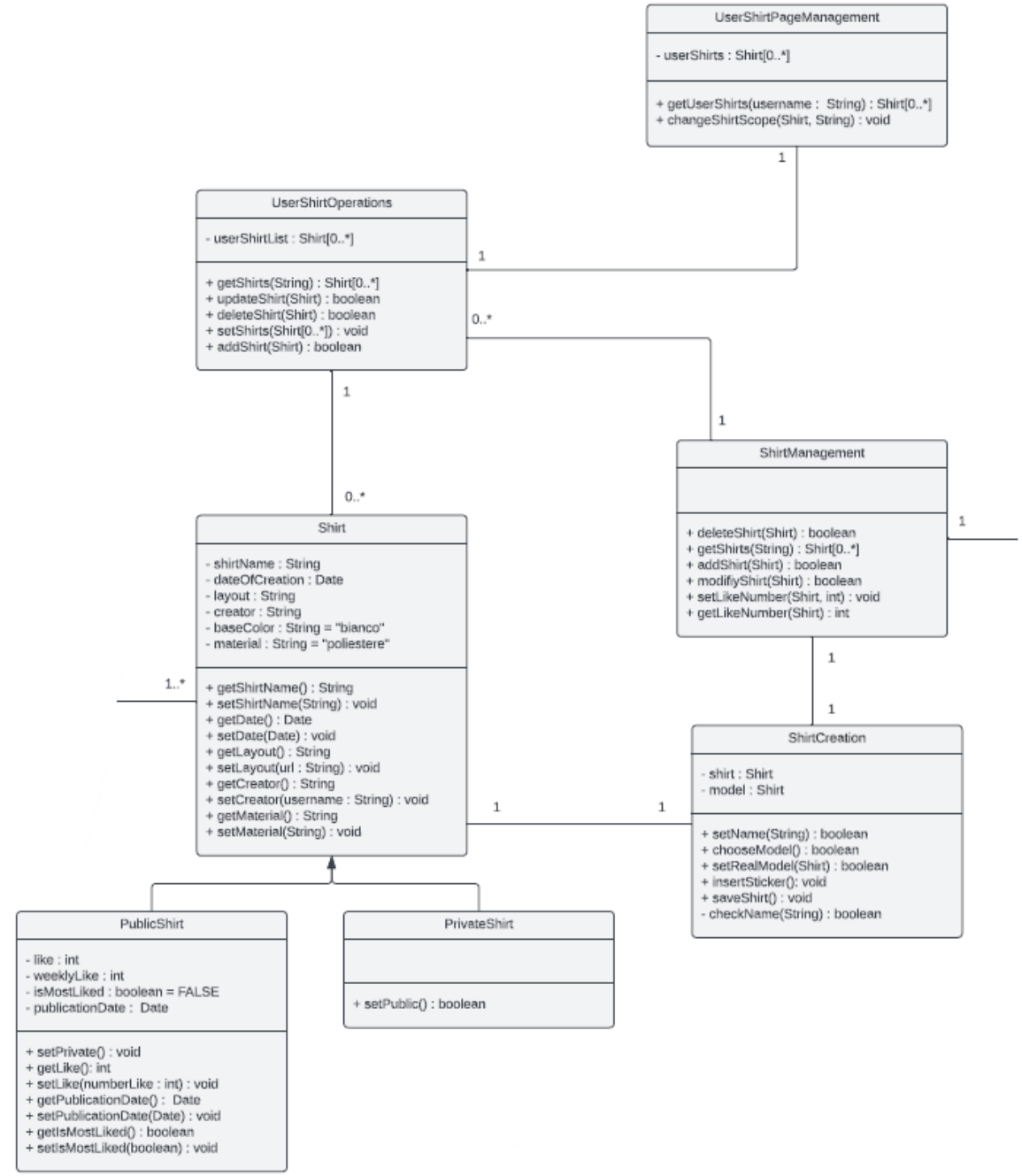
Gestione acquisti

Dal diagramma dei componenti è stata identificata una classe *ShoppingCart*, che si occupa di gestire il carrello associato ad un utente. Questa classe prevede la visualizzazione delle magliette presenti al suo interno, selezionate precedentemente dall'utente, per poter procedere all'acquisto o alla modifica di alcuni parametri quali quantità e materiale. Le magliette possono anche essere rimosse.
Se l'utente procede all'acquisto la classe comunicherà alla classe *PaymentManagement* i dati necessari (dati utente, dati dell'ordine e prezzo totale) per poter indirizzare l'utente verso il pagamento. Una volta che il pagamento è stato effettuato, *PaymentManagement* comunica l'ordine a *OrderManagement*, che lo memorizzerà nel database per poterne permettere la visualizzazione in un secondo momento. In caso di pagamento non riuscito l'ordine verrà comunque memorizzato ma contrassegnato con pagamento non riuscito. L'interfaccia ordini, identificata con la classe *OrderPageManagement*, permette all'utente tali interazioni con il proprio storico degli ordini.



Gestione magliette utente

Dal diagramma dei componenti è stata identificata una classe *UserShirtOperations*, che permette all'utente di effettuare tutte le operazioni riguardo le proprie magliette quali visualizzazione, modifica, cancellazione e creazione (CRUD).
Nel caso di creazione o modifica si visualizza la pagina dedicata, gestita dalla classe *ShirtCreation*.
Nel caso, invece, di cancellazione si invia una richiesta di "*delete*".
Per effettuare attivamente le operazioni a database troviamo la classe *ShirtManagement*, che oltre alle operazioni CRUD sopra citate si occupa anche di memorizzare i *like* delle magliette e recuperarli per la visualizzazione in front-end. Tramite la classe *UserShirtPageManagement*, l'utente può modificare la visibilità delle proprie magliette, rendendole pubbliche o private.



Gestione filtri

Dal diagramma dei componenti è stata identificata una classe *FilterManagement*, che permette l'ordinamento e il filtraggio durante la visualizzazione delle magliette. Questa viene utilizzata sia per il recupero delle magliette da esporre nella homepage, tramite la comunicazione con la classe *HomePageManagement*, che per il recupero delle magliette in entrambe le sezioni (*magliette dell'azienda* e magliette della community), rispettivamente gestite dalle classi *CompanyPageManagement* e *CommunityPageManagement*.

Una barra di ricerca, con relativa classe *SearchBar*, permette all'utente di effettuare ricerche personalizzate, inserendo le parole chiave che identificano il prodotto desiderato.

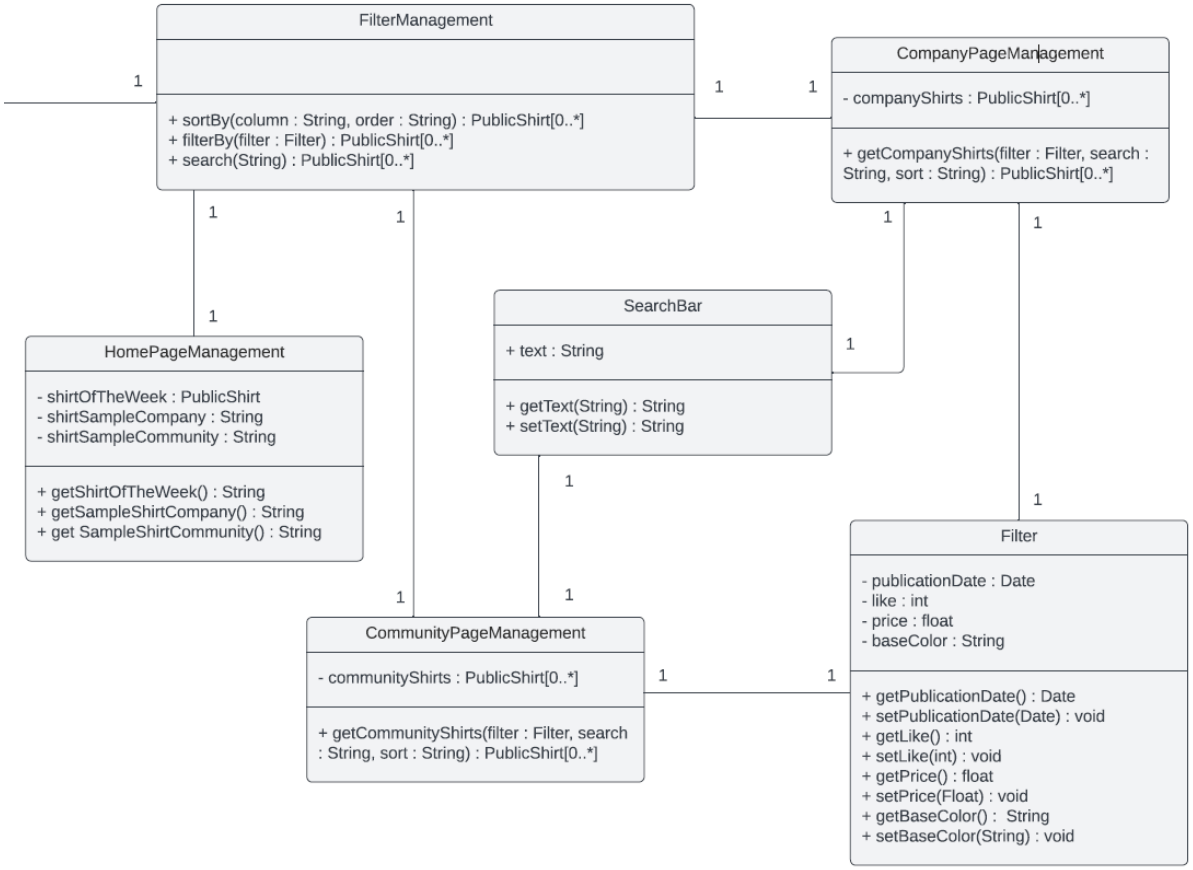


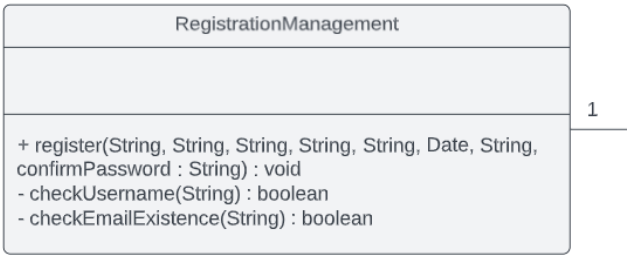
Diagramma delle Classi

[Schema in SVG]

Codice in Object Constraint Language

Gestione registrazione

Le seguenti condizioni si applicano sulla classe *RegistrationManagement.*



Nel form di registrazione, viene richiesto obbligatoriamente di inserire la password in due campi diversi:

- Un campo in cui viene richiesto l'inserimento della nuova password
- Un campo in cui viene richiesto di riscrivere la stessa password appena scelta, per confermare che non ci siano errori di battitura nella prima.

Il form, quindi, è completabile solo se la password coincide in entrambi i campi. Ciò viene espresso in OCL attraverso una precondizione scritto nella forma seguente:

```
context RegistrationManagement :: register(String, String, String,
String, String, Date, String password, String confirmPassword)
pre : password = confirmPassword
```

La password deve inoltre rispettare le regole di formattazione imposte nel punto **RNF 4:**

- lunghezza di almeno 8 caratteri
- almeno una lettera minuscola
- almeno una lettera maiuscola
- almeno un numero
- almeno un carattere speciale tra quelli individuati nella tabella ASCII nell'intervallo [33, 47]

In OCL viene rappresentato mediante la precondizione:

```
context RegistrationManagement :: register(String, String, String, String, String,
Date, String password, String)
pre : password.exists(c | c =
"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$")
```

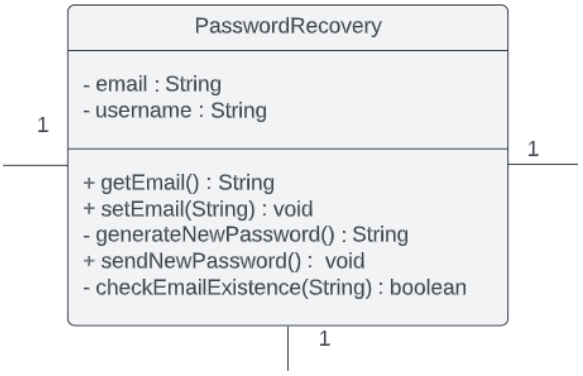
Come specificato nel punto **RNF 8**, username e email devono essere univoci, ovvero non essere già associati ad un altro account. Ciò viene espresso in OCL tramite la precondizione seguente:

```
context RegistrationManagement :: register(String username,
String, String, String, String, Date, String, String)
pre : checkUsername(username) = FALSE
```

Recupero della password

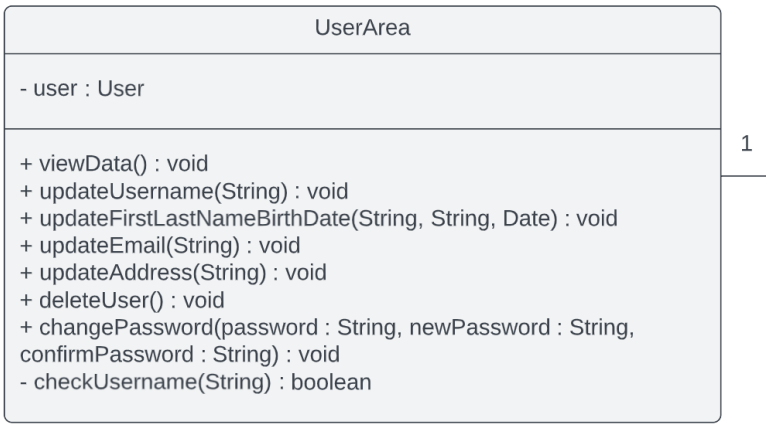
Il recupero della password richiede all'utente l'inserimento della propria e-mail. Per evitare di inviare password a indirizzi e-mail non corretti, la e-mail inserita dall'utente deve essere presente nel database ed essere associata ad un utente.
Ciò viene descritto in OCL dalla precondizione:

```
context PasswordRecovery :: sendNewPassword()
pre : checkMailExistence(email) = TRUE
```



Modifica dati utente

La classe *UserArea* contiene metodi adibiti alla modifica degli attributi della classe *Utente*, della password e l'eliminazione del profilo dal database.



L'utente può modificare la propria password nella sezione 'Area utente'. Oltre all'inserimento della vecchia password e la verifica della sua correttezza, l'utente deve compilare la sezione di modifica vera e propria: due campi di inserimento password:

- La prima richiede la nuova password
- La seconda richiede il reinserimento della stessa password appena scelta

I due campi devono avere lo stesso contenuto perché il cambio possa procedere correttamente. In OCL, viene descritto attraverso la precondizione seguente:

```
context UserArea :: changePassword(String password, String newPassword, String
confirmPassword)
pre : newPassword = confirmPassword
```

La nuova password deve rispettare dei criteri di formattazione, descritti anche nel paragrafo precedente e specificato nell'analisi dei requisiti al punto **RNF 4**, quali:

- lunghezza di almeno 8 caratteri
- almeno una lettera minuscola
- almeno una lettera maiuscola
- almeno un numero
- almeno un carattere speciale tra quelli individuati nella tabella ASCII nell'intervallo [33, 47]

La descrizione di questi requisiti in OCL si imposta con la seguente precondizione:

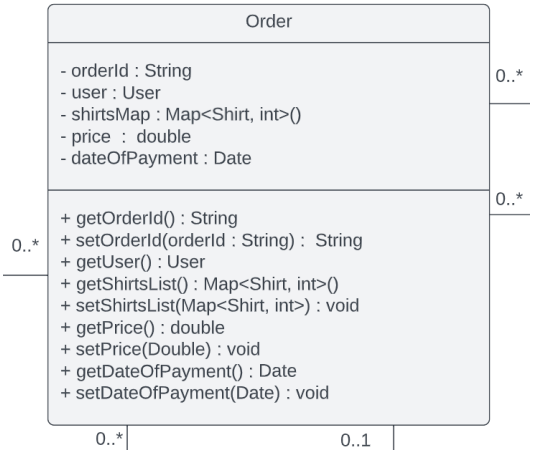
```
context UserArea :: changePassword(String password, String newPassword, String
confirmPassword)
pre : newPassword.exists(c | c =
"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$")
```

Se l'utente modifica il suo username viene effettuato un controllo sulla nuova scelta, in quanto lo username deve essere necessariamente univoco. La precondizione viene rappresentata in OCL in questo modo:

```
context UserArea :: updateUsername(String)
pre : checkUsername(newUsername) = FALSE
```

Ordine

Un'istanza della classe *Ordine* può esistere solamente se rispetta una serie di condizioni.



Un ordine può essere effettuato solo se l'attributo *shirtsMap* non è vuota, che viene descritto in OCL attraverso l'invariante seguente:

```
context Order
inv : shirtsMap.count() > 0
```

Una conseguenza della condizione precedente implica che l'attributo *price* sia maggiore di 0 e ciò viene espresso in OCL con la seguente invariante:

```
context Order
inv : price > 0
```

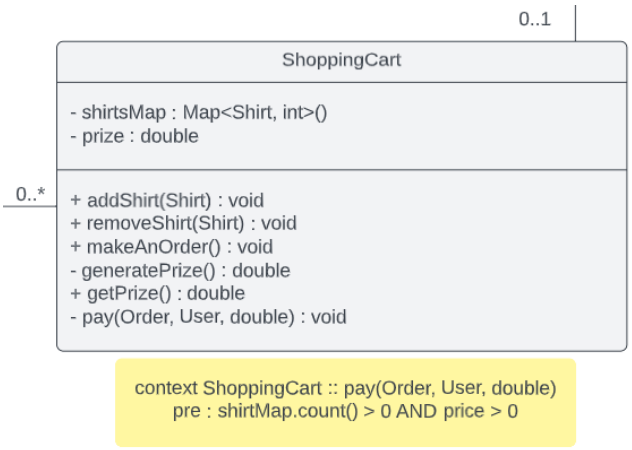
Per evitare eventuali errori nell'inserimento della data, *dateOfPayment* deve essere antecedente o corrispondente alla data attuale. La descrizione di questa condizione in OCL avviene attraverso un invariante:

```
context Order
inv : dateOfPayment <= CURRENT_DATE
```

La lista e il prezzo delle magliette si ricevono dal carrello nel momento in cui viene effettuato un acquisto, ma si decide di controllarle comunque per evitare la possibilità di presentare incongruenze nei dati.

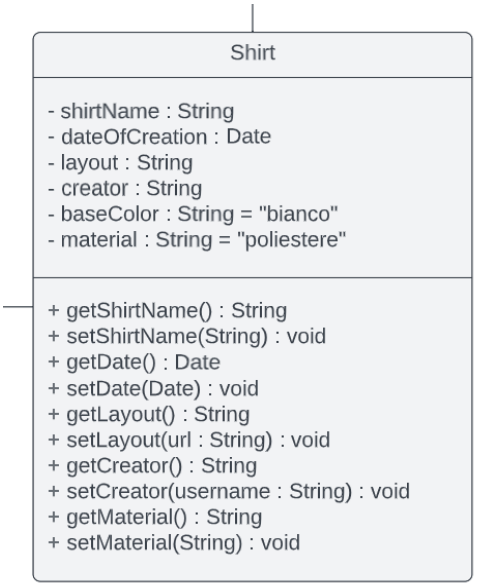
Carrello

Un acquisto può essere completato solo se nel carrello è presente almeno una maglietta, *shirtsMap* deve contenere almeno un elemento, e l'attributo price deve essere maggiore di 0. Ciò viene rappresentato in OCL dalla seguente invariante:



Maglietta

Le seguenti condizioni si applicano alla classe Shirt, che rappresenta l'oggetto maglietta.



Il materiale di una maglietta, definito da *material*, deve essere sempre assegnato ad un valore: tra sintetico (poliestere), rayon, cotone e organico. Viene impostato di default il materiale sintetico e modificato successivamente dall'utente. Ciò viene descritto in OCL attraverso l'invariante:

```
context Shirt
inv : material = "poliestere" OR material = "cotone" OR
material = "organico" OR material = "rayon"
```

Il colore della maglietta, identificato dall'attributo *baseColor*, deve essere obbligatoriamente associato ad un valore tra i seguenti:

- bianco
- nero
- rosso
- arancione
- giallo
- verde
- blu
- rosa

Viene impostato il colore 'bianco' di default e può essere modificato successivamente dall'utente.
Questa condizione è espressa in OCL attraverso un invariante con la seguente forma:

```
context Shirt
pre : color = "bianco"
inv : color = "bianco" OR color = "nero" OR color = "rosso"
OR color = "arancione" OR color = "giallo" OR color =
"verde" OR color = "blu" OR color = "rosa"
```

Magliette pubbliche

Le condizioni appena descritte si applicano alla classe *PublicShirt*.

Le magliette pubbliche presentano un numero di “mi piace” totale, identificato dall’attributo like, che deve essere sempre maggiore o uguale al numero dei “mi piace” settimanali, rappresentato dall’attributo *weeklyLike*. Questa condizione si descrive in OCL con la seguente invariante:

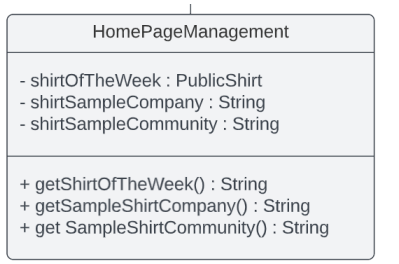
```
context PublicShirt
inv : like >= weeklyLike
```

Presentano poi un attributo per contrassegnare la maglietta della settimana, *isMostLiked*, impostato di default a *false*, per identificare la maglietta della settimana (che ha *isMostLiked* impostato a *true*).
publicationDate, che rappresenta la data di pubblicazione della maglietta, deve essere sempre antecedente o corrispondente a quella corrente e sempre conseguente o corrispondente alla data di creazione, rappresentato dall’attributo *dateOfCreation*. Questa condizione viene rappresentato in OCL con la seguente invariante:

```
context PublicShirt
inv : publicationDate <= CURRENT_DATE
AND publicationDate >= dateOfCreation
```

Maglietta della settimana

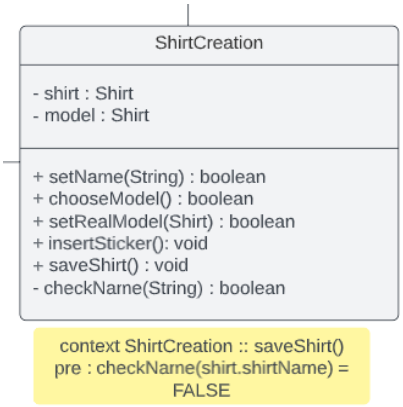
L’homepage mette in evidenza la maglietta della settimana, ovvero quella che ha ricevuto più likes nella settimana precedente. Questa maglietta avrà il flag *isMostLiked* impostato a true. Ciò viene descritto in OCL dalla seguente invariante:




```
context HomePageManagement
inv : shirtOfTheWeek.getIsMostLike() = TRUE
```

Creazione della maglietta

Per poter salvare una maglietta creata o modificata è necessario scegliere un nome per la maglietta, ovvero non lasciare il campo *shirtName* dell’attributo *shirt* vuoto. In OCL viene descritto attraverso la seguente precondizione:



Ordinamento visualizzazione magliette

Le magliette si possono visualizzare in modo ordinato. Viene gestito dal metodo *sortBy* che prende in input il criterio di ordinamento: date di pubblicazione (*dateOfPublication*), nomi delle magliette (*shirtName*), prezzo (*price*) e numero di like (*like*), e prende l'ordine di visualizzazione, ascendente o discendente. Le condizioni dell'input sono espresso in OCL attraverso le seguenti precondizioni:

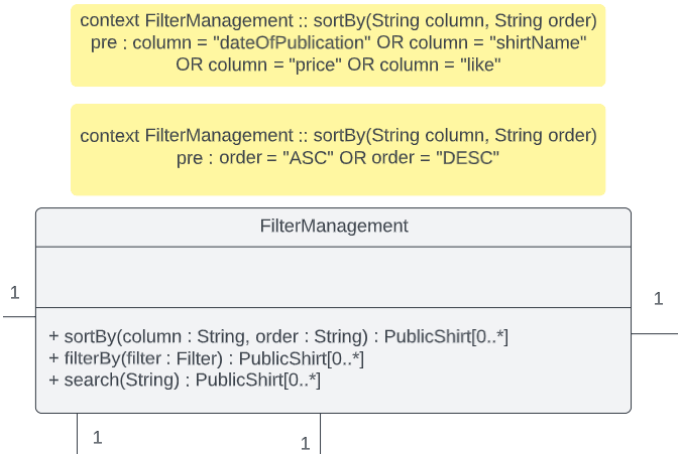
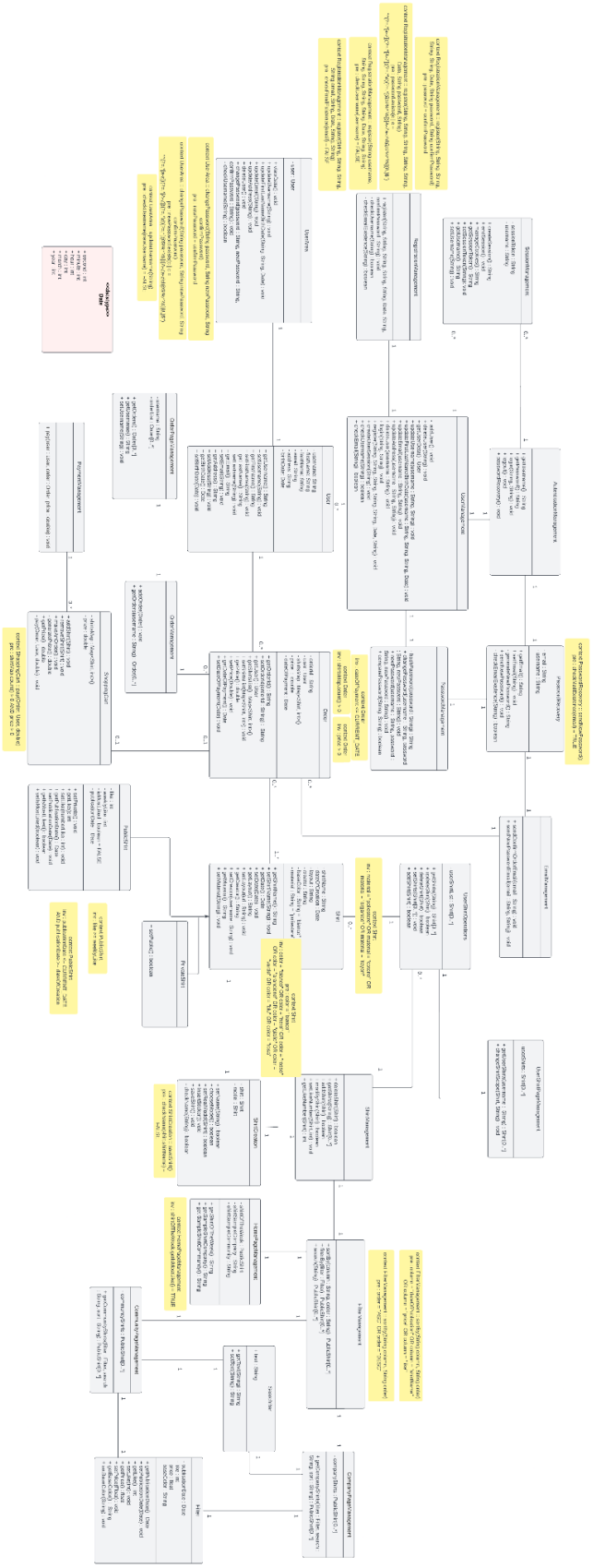


Diagramma delle Classi in OCL



Schema in SVG