

Progetto Ingegneria del Software - eCommerce

Colafranceschi Luca 1963232

Ippoliti Beatrice 1982749

13 giugno 2024

Sommario

Your abstract.

1 Introduzione

Il seguente progetto rappresenta una piattaforma e-commerce che consente agli utenti di interagire con il sistema con la possibilità di inserire nuovi prodotti disponibili alla vendita, acquistarli e gestire la loro spedizione.

Il sistema è composto da tre server che gestiscono le richieste di *Customer*, *Vendor* e *Transporter*, implementato in C++ e provvisto di un Database PostgreSQL.

Gli utenti, tramite il Client, possono utilizzare vari comandi e interagire con il sistema. Il Client e il Server comunicano tramite il servizio di messaggistica *Redis*.

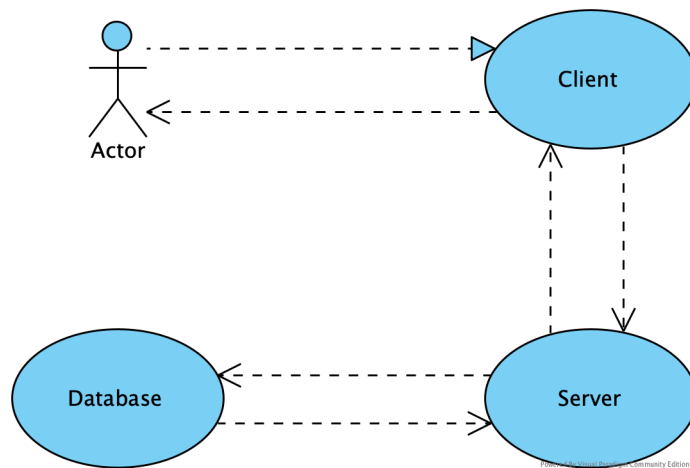


Figura 1: Ambiente

1.1 Accesso al sistema

Al momento del primo accesso al sistema viene richiesta la registrazione da parte dell'utente con le seguenti informazioni:

- Username
- Email
- Password

- Indirizzo di spedizione (richiesta solo per gli utenti Customer)

A seguito della registrazione l'utente può, in un secondo momento, accedere al sistema fornendo username e password per effettuare il login.

1.2 Descrizione generale

Il sistema permette agli utenti Vendor di aggiungere i propri prodotti al Database, specificando nome e prezzo unitario.

Gli utenti Customer possono effettuare la ricerca di un prodotto attraverso il nome e acquistarne una quantità arbitraria.

Gli utenti Transporter possono fare richiesta al sistema in modo che gli venga assegnato il trasporto di un ordine disponibile alla spedizione. Inoltre è possibile modificare lo stato delle spedizioni a proprio carico.

2 Requisiti sui dati

Customer:

- Id
- Username
- Email
- Password
- Shipping address

Vendor:

- Id
- Username
- Email
- Password

Transporter:

- Id
- Username
- Email
- Password

Insertion:

- Id
- Name
- Price
- Instant
- Insertion status
- Vendor id

Orders:

- Id
- Product id
- Quantity
- Customer id
- Instant date
- Assigned

Shipping:

- Id
- Order id
- Transporter id
- Shipping status
- Delivered

3 User Requirements

Di seguito una lista dei comandi che possono essere utilizzati dai vari utenti del sistema.

3.1 Utente non registrato

- Registration: l'utente deve inserire email, username e password per registrarsi;

3.2 Customer

- Registration: l'utente deve inserire email, username, password e indirizzo di spedizione per registrarsi;
- Login: l'utente può accedere al sistema fornendo username e password inseriti in fase di registrazione;
- Search Product: l'utente può, inserendo il nome del prodotto desiderato, visualizzare una lista dei prodotti che includono il nome fornito.
- Purchase Product: dopo la ricerca, l'utente può selezionare un prodotto e procedere con l'acquisto, specificando la quantità desiderata.
- Visualize Orders: l'utente può visualizzare tutti gli ordini effettuati con il relativo stato dell'ordine (pending, delivering, delivered).
- Quit: l'utente può uscire dal sistema.

3.3 Vendor

- Login: l'utente può accedere al sistema fornendo username e password inseriti in fase di registrazione;
- New Insertion: l'utente può inserire un nuovo prodotto nel sistema specificandone il nome e il prezzo;
- Visualize Insertion: l'utente può visualizzare i prodotti inseriti nel sistema e le relative vendite di quel prodotto;
- Quit: l'utente può uscire dal sistema.

3.4 Transporter

- Login: l'utente può accedere al sistema fornendo username e password inseriti in fase di registrazione;
- New Shipping: l'utente può fare richiesta per l'assegnazione di una nuova spedizione;
- Visualize Shipping: l'utente può visualizzare le spedizioni che ha preso in carico;
- Modify Shipping Status: l'utente può modificare lo stato di spedizione dell'ordine del customer;
- Quit: l'utente può uscire dal sistema.

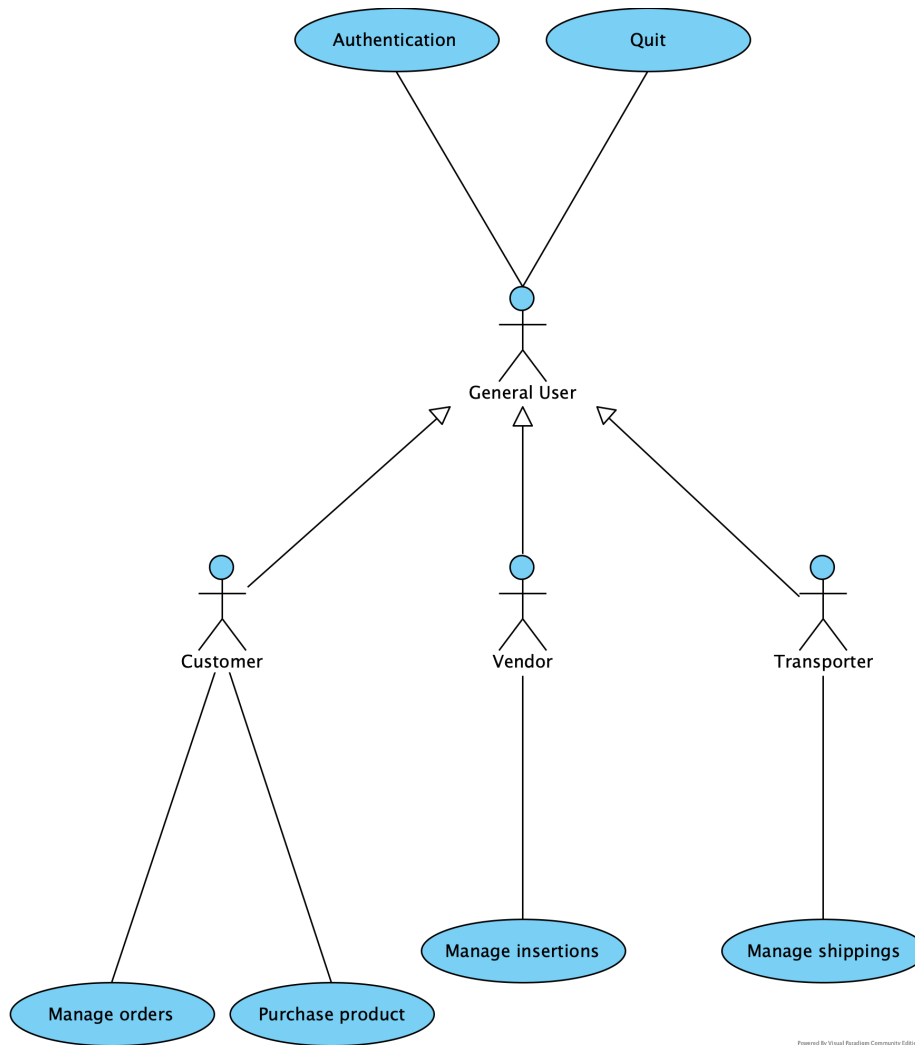


Figura 2: Ambiente

4 System Requirements

4.1 Registration

Dopo che l'utente ha fornito le credenziali, vengono salvate le informazioni; se esiste già un utente registrato con l'email fornita, il sistema rifiuta la richiesta di registration.

4.2 Login

L'utente fornisce username e password per effettuare il login; se nel database non esiste l'utente con le credenziali fornite, il sistema rifiuta l'operazione.

4.3 New insertion

Il vendor fornisce il nome del nuovo prodotto e il prezzo e richiede al sistema di inserire la nuova insertion. Questa viene creata nel database con insertion status "available".

4.4 Manage insertions

Il vendor può visualizzare gli ordini effettuati relativi ai suoi prodotti con il numero di vendite associato.

4.5 New order

Il customer specifica il prodotto e la quantità e il nuovo ordine viene creato nel database.

4.6 Manage order

Il customer può visualizzare lo stato di tutti gli ordini effettuati.

4.7 New shipping

Il transporter fa richiesta al sistema per l'assegnazione di una nuova spedizione. Se esiste un ordine disponibile per la spedizione, allora viene creata nel database la nuova spedizione con il primo ordine in coda.

4.8 Manage shipping

Il transporter fa richiesta al sistema per modificare lo stato della spedizione quando questo viene consegnato al customer.

4.9 Activity Diagram UML

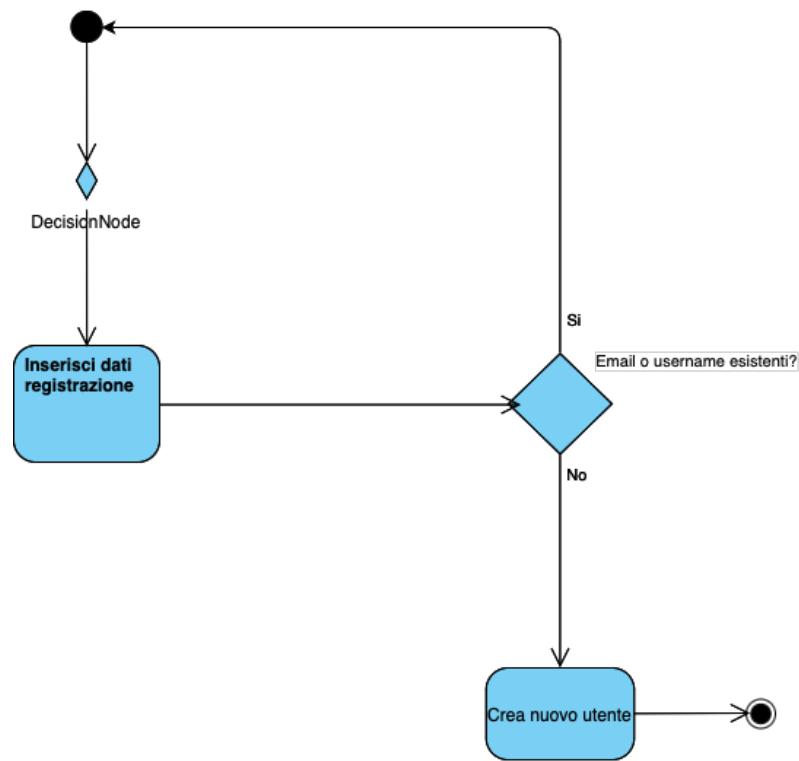


Figura 3: Activity Diagram UML - Operazione Registrazione

4.10 State Diagram UML

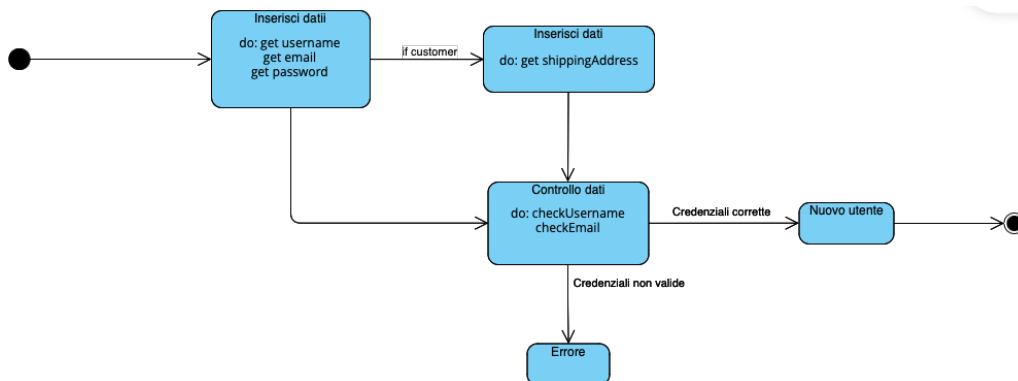


Figura 4: State Diagram UML - Registrazione

4.11 Message Sequence Chart UML

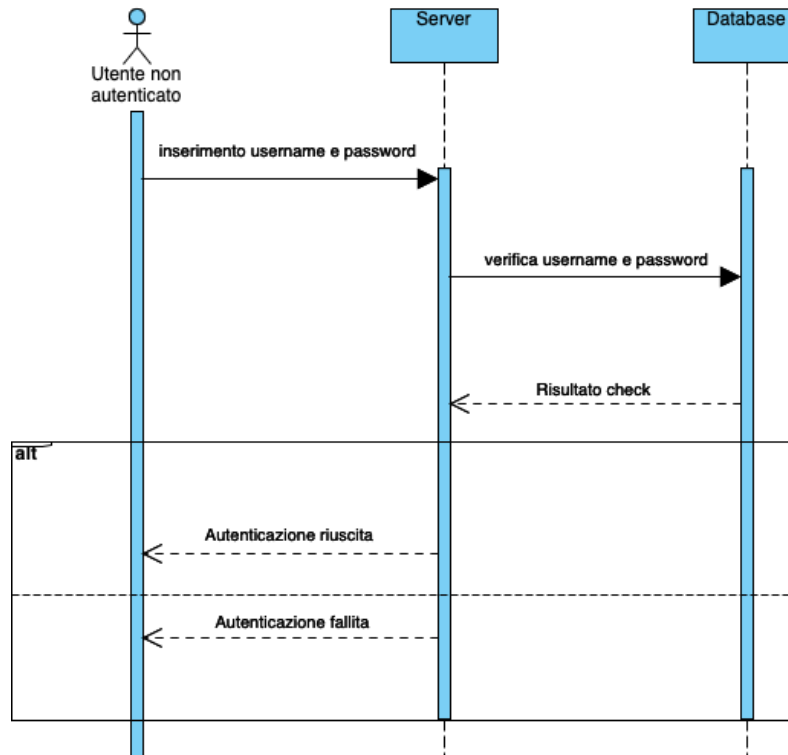


Figura 5: Message Sequence Chart UML - Login

5 Implementation

Gli attori del nostro sistema sono:

- **Vendor:** le sue funzioni sono le seguenti:
 - *Registration:* invia una query al Database ricevuta dal lato client tramite messaggistica Redis. La query contiene le informazioni per la registrazione. Se nel Database è già presente un utente con la stessa email, allora il sistema rifiuta la registrazione e invia al client un messaggio di errore.
 - *Login:* invia una query al database con username e password ricevuti dal client tramite messaggistica Redis. In assenza di match delle credenziali all'interno del database, il sistema rifiuta la richiesta e invia un messaggio di errore al client. Altrimenti, viene restituito l'id utente che sarà inviato al client insieme al messaggio di operazione riuscita. In questo modo il client ha accesso al sistema.
 - *NewInsertion:* a seguito della registrazione o del login, viene inviata una query al database, ricevuta dal client tramite messaggistica Redis, con le informazioni relative al nuovo prodotto da inserire nel sistema. Se c'è un errore durante l'esecuzione della query, viene ritornato un messaggio di errore al client.
 - *ManageInsertion:* invia la query al database per ricevere le informazioni riguardo tutte le inserzioni dell'utente richiedente.
- **Customer:** le sue funzioni sono le seguenti:
 - *Registration:* invia una query al Database ricevuta dal lato client tramite messaggistica Redis. La query contiene le informazioni per la registrazione. Se nel

Database è già presente un utente con la stessa email, allora il sistema rifiuta la registrazione e invia al client un messaggio di errore.

- *Login*: invia una query al database con username e password ricevuti dal client tramite messaggistica Redis. In assenza di match delle credenziali all'interno del database, il sistema rifiuta la richiesta e invia un messaggio di errore al client. Altrimenti, viene restituito l'id utente che sarà inviato al client insieme al messaggio di operazione riuscita. In questo modo il client ha accesso al sistema.
- *NewOrder*: a seguito della registrazione o del login, viene inviata una query al database, ricevuta dal client tramite messaggistica Redis. La query contiene le informazioni necessarie per inserire il nuovo ordine all'interno del database. Se c'è un errore durante l'esecuzione della query, viene inviato un messaggio di errore al client, altrimenti viene notificato il successo dell'operazione.
- *ManageOrder*: invia una query al database per ricevere le informazioni riguardanti tutti gli ordini dell'utente richiedente.

• **Transporter**: le sue funzioni sono le seguenti:

- *Registration*: invia una query al Database ricevuta dal lato client tramite messaggistica Redis. La query contiene le informazioni per la registrazione. Se nel Database è già presente un utente con la stessa email, allora il sistema rifiuta la registrazione e invia al client un messaggio di errore.
- *Login*: invia una query al database con username e password ricevuti dal client tramite messaggistica Redis. In assenza di match delle credenziali all'interno del database, il sistema rifiuta la richiesta e invia un messaggio di errore al client. Altrimenti, viene restituito l'id utente che sarà inviato al client insieme al messaggio di operazione riuscita. In questo modo il client ha accesso al sistema.
- *NewShipping*: dopo aver ricevuto la richiesta dal client, tramite messaggistica Redis, invia al database una query per richiedere un ordine disponibile per la spedizione. Se vengono ritornati più ordini, sarà assegnato alla spedizione quello meno recente. Se invece non ci sono ordini disponibili per la spedizione, verrà inviato al client un messaggio con il codice che indica la mancata disponibilità degli ordini.
- *ChangeShippingStatus*: dopo aver ricevuto la richiesta dal client, tramite messaggistica Redis, contenente il codice della spedizione e il nuovo stato, viene eseguita una query nel database per modificare lo stato della spedizione con quello appena ricevuto. Se durante l'esecuzione della query, si verifica un errore, allora viene inviato un messaggio di errore al client tramite Redis.
- *ManageShipping*: invia la query al database per ricevere le informazioni riguardo tutte le spedizioni dell'utente richiedente

5.1 Database

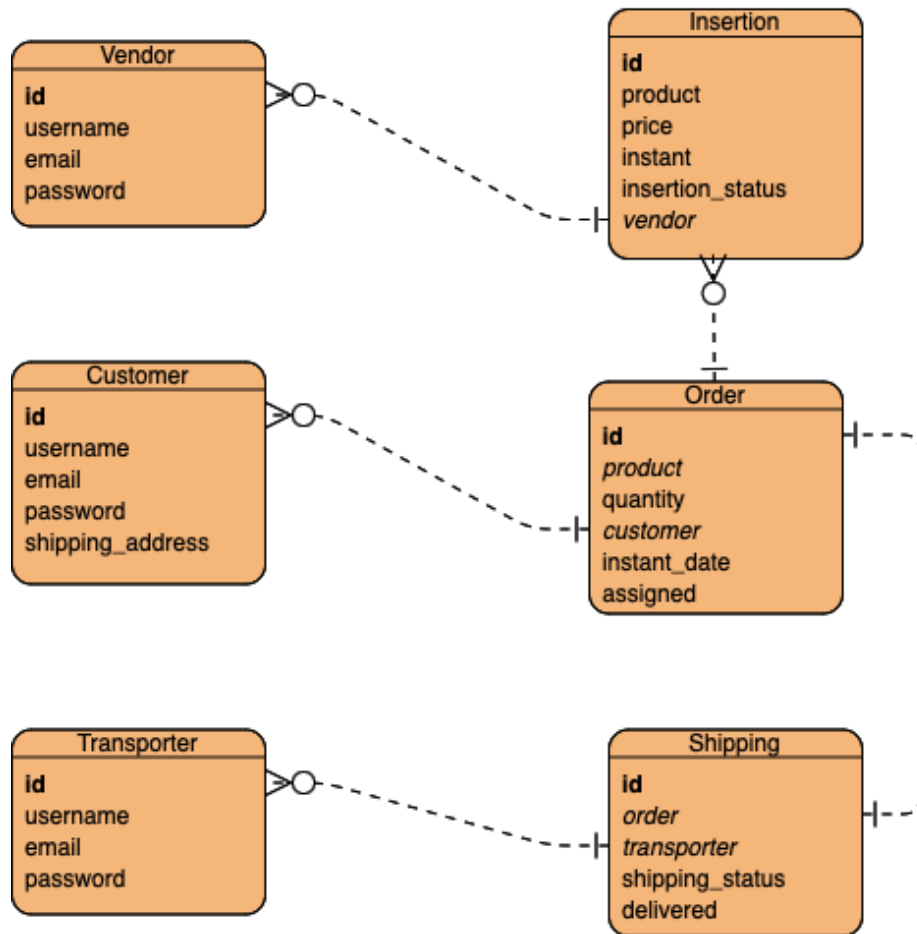


Figura 6: Database

- **Table Vendor**: rappresenta l'utente Vendor in cui vengono memorizzati id, username, email, password; non possono esserci valori ripetuti nella tabella per gli attributi username ed email.
- **Table Customer**: rappresenta l'utente Customer in cui vengono conservate id, username, email, password, shipping_address e data iscrizione. Non possono esserci più utenti che hanno lo stesso username ed email.
- **Table Transporter**: rappresenta l'utente Transporter in cui vengono memorizzati id, username, email, password; non possono esserci più transporter che hanno lo stesso username ed email.
- **Table Insertion**: rappresenta le Insertion, cioè i prodotti creati dal Vendor per essere venduti e acquistati dai Customer. Vengono conservate id, product, price, instant, insertion_status, vendor (foreign key che rappresenta l'id del vendor che ha creato l'insertion).
- **Table Orders**: rappresenta gli Orders, ovvero i prodotti che vengono acquistati dal Customer. Vengono memorizzati id, quantity, customer (foreign key che rappresenta l'id del customer che ha effettuato l'ordine), instant_date (istante in cui viene

effettuato l'ordine) e assigned (booleano che indica se l'ordine è stato assegnato a un Transporter per la spedizione).

- **Table Shipping:** rappresenta le Shipping, cioè le spedizioni relative agli ordini effettuati che vengono assegnate a un Transporter per la spedizione. Vengono conservati id, order (foreign key che rappresenta l'id dell'ordine), transporter (foreign key che rappresenta l'id del Transporter a cui è assegnata la spedizione), shipping_status e delivered (booleano che indica se la spedizione è stata consegnata).

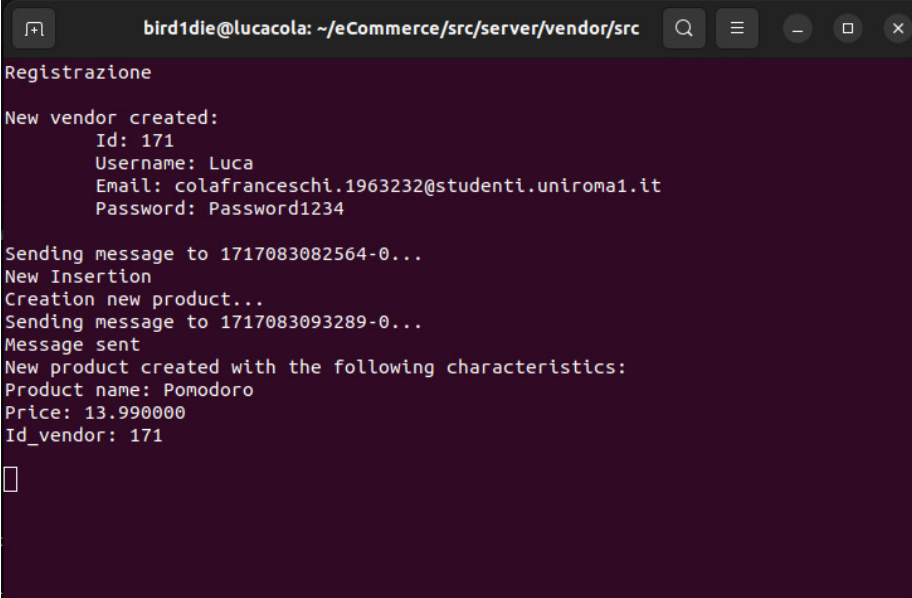
5.2 Redis

La figura 7 rappresenta il processo di registrazione e creazione dell'*Insertion* da parte del Vendor. Dal lato client viene mandato un messaggio sul canale di stream *vendor*. I messaggi sono formattati nel seguente modo

```
"XADD vendor * operation_id 2 product_name %s price %s id_vendor %s"
```

```
"XADD customer * operation_id 1 username %s email %s password %s  
shipping_address %s"
```

Dal lato server, che è sempre in ascolto, il messaggio viene formattato in base al codice assegnato alla *operation_id* e vengono eseguite le operazioni di registrazione, che ritorna l'id dell'utente appena creato, e della creazione dell'insertion. Infine viene rimandato al client un messaggio con l'esito dell'operazione.



```
bird1die@lucacola: ~/eCommerce/src/server/vendor/src
Registrazione
New vendor created:
  Id: 171
  Username: Luca
  Email: colafranceschi.1963232@studenti.uniroma1.it
  Password: Password1234
Sending message to 1717083082564-0...
New Insertion
Creation new product...
Sending message to 1717083093289-0...
Message sent
New product created with the following characteristics:
Product name: Pomodoro
Price: 13.990000
Id_vendor: 171
```

Figura 7: Vendor operation

La figura 8 rappresenta il processo di registrazione e creazione dell'ordine da parte dell'utente Customer. Dal lato client viene inviato un messaggio sul canale di stream *customer*. I messaggi sono formattati nel seguente modo:

```
"XADD customer * operation_id 1 username %s email %s password %s  
shipping_address %s"
```

```
"XADD customer * operation_id 2 id_product %s quantity %s id_customer %s"
```

```
bird1die@lucacola: ~/eCommerce/src/server/customer/src
Registrazione
Username: LucaCliente
Email: colafranceschi.1963232@studenti.uniroma1.it
Password: Pass1234
New customer created with the following characteristics:
Username: LucaCliente
Email: colafranceschi.1963232@studenti.uniroma1.it
Password: Pass1234

Sending message to 1717083295765-0...
Message sent
New Order
Creating new order...
New order created with:
Id: 46
Quantity: 2
Id_customer: 90

Sending message to 1717083307299-0...
Message sent
█
```

Figura 8: Customer operation

Dal lato server, che rimane in ascolto, il messaggio viene formattato in base al codice assegnato alla *operation_id* e in seguito vengono eseguite le operazioni di registrazione e creazione dell'ordine. Infine viene inviato al client un messaggio con l'esito dell'operazione.

Nella figura 9 è rappresentato il processo della creazione di una nuova spedizione a carico del Transporter e registrazione al sistema da parte dello stesso. I messaggi di richiesta vengono inviati al server sul canale stream *transporter*. Il pattern dei messaggi è il seguente

```
"XADD transporter * operation_id 4 id_transporter %s"
```

```
"XADD transporter * operation_id 1 username %s email %s password %s"
```

```
bird1die@lucacola: ~/eCommerce/src/server/trasporter/src
New shipping created

Sending message to 1717083466132-0...
Message sent
Registrazione
Creating new transporter within the database...
New transporter created with:
Username: BeaTrasportatrice
Email: beatrice.1982749@studenti.uniroma1.it
Password:

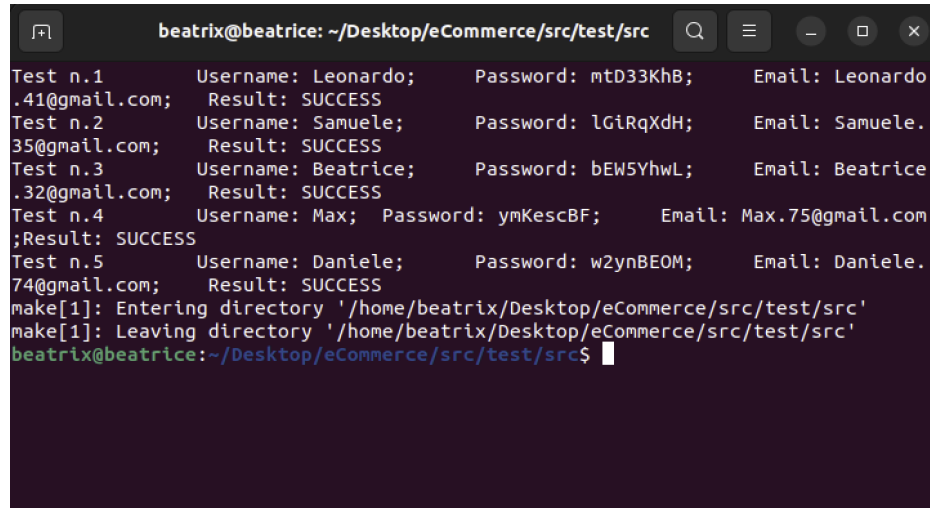
Sending message to 1717083503255-0...
Message sent
█
```

Figura 9: Transporter operation

Il server cerca l'ordine effettuato da più tempo che non è stato ancora assegnato a una spedizione. Successivamente manda un messaggio al client con l'esito dell'operazione.

6 Test

Per effettuare i test vengono generati randomicamente i dati per effettuare la registrazione dell'utente vendor. L'*username* viene preso da un file in cui sono elencati dei nomi, la *password* invece viene generata randomicamente utilizzando caratteri alfanumerici.

A terminal window titled 'beatrix@beatrice: ~/Desktop/eCommerce/src/test/src' displays the output of a test script. The script performs five tests (Test n.1 to Test n.5), each generating a random username, password, and email, and then attempting to register the user. All five tests result in 'SUCCESS'. The terminal output is as follows:

```
Test n.1      Username: Leonardo;      Password: mtD33KhB;      Email: Leonardo
.41@gmail.com; Result: SUCCESS
Test n.2      Username: Samuele;      Password: lGiRqXdH;      Email: Samuele.
35@gmail.com; Result: SUCCESS
Test n.3      Username: Beatrice;      Password: bEW5YhWL;      Email: Beatrice
.32@gmail.com; Result: SUCCESS
Test n.4      Username: Max;      Password: ymKescBF;      Email: Max.75@gmail.com
;Result: SUCCESS
Test n.5      Username: Daniele;      Password: w2ynBEOM;      Email: Daniele.
74@gmail.com; Result: SUCCESS
make[1]: Entering directory '/home/beatrix/Desktop/eCommerce/src/test/src'
make[1]: Leaving directory '/home/beatrix/Desktop/eCommerce/src/test/src'
beatrix@beatrice:~/Desktop/eCommerce/src/test/src$
```