



Università Ca' Foscari - Venezia

Relazione del Progetto di Basi di Dati

Piattaforma di E-commerce

Massimo Costantini - 877336

Samuel Obeng Takyi - 881431

Filippo Tiozzo - 887683

Anno Accademico 2023 - 2024

<b>Indice</b>	<b>2</b>
<b>1     Introduzione</b>	<b>3</b>
1.1     Applicazione	3
1.2     Blueprint	3
1.3     Caricamento grant e trigger	3
<b>2     Funzionalità principali</b>	<b>4</b>
2.1     Esecuzione dell'applicazione	5
2.2     Requisiti del progetto	12
<b>3     Progettazione concettuale e logica della basi di dati</b>	<b>13</b>
3.1     Modello Grafico ad Oggetti	13
3.2     Tabelle	14
3.3     Relazioni	18
3.4     Modello Logico Relazionale	19
<b>4     Query principali</b>	<b>20</b>
<b>5     Principali scelte progettuali</b>	<b>21</b>
5.1     Autorizzazioni	22
5.2     Trigger	23
<b>6     Ulteriori informazioni</b>	<b>25</b>
6.1     Comandi utili	25
<b>7     Contributo al progetto</b>	<b>27</b>

# **1 Introduzione**

Il progetto scelto riguarda la Piattaforma di E-commerce, una web application sviluppata in linguaggio Python mediante il framework Flask, ed interfacciata con il database relazionale PostgreSQL utilizzando la libreria SQLAlchemy.

In questo documento sono descritte le funzionalità principali dell'applicazione, la progettazione concettuale e logica della base di dati, le query e le scelte progettuali che sono state fatte.

## **1.1 Applicazione**

L'interfaccia web permette di registrarsi e collegarsi come acquirente oppure come venditore fornendo una serie di possibili azioni, diverse a seconda del ruolo, come ad esempio la scelta da parte dell'acquirente dei prodotti da acquistare e la creazione da parte del venditore di nuovi prodotti da vendere.

## **1.2 Blueprint**

L'applicazione è suddivisa in Blueprint Flask, un insieme di route, cioè di percorsi URL, che gestiscono le chiamate HTTP.

In questo modo il codice sorgente è suddiviso in moduli (i file `acquirente.py`, `logica.py`, `modelli.py` e `venditore.py`) che facilitano così la gestione del progetto.

## **1.3 Caricamento grant e trigger**

Per il caricamento dei grant e dei trigger necessari al funzionamento dell'applicazione, sono stati inseriti all'interno del progetto due file chiamati `grant.sql` e `trigger.sql`.

## 2 Funzionalità principali

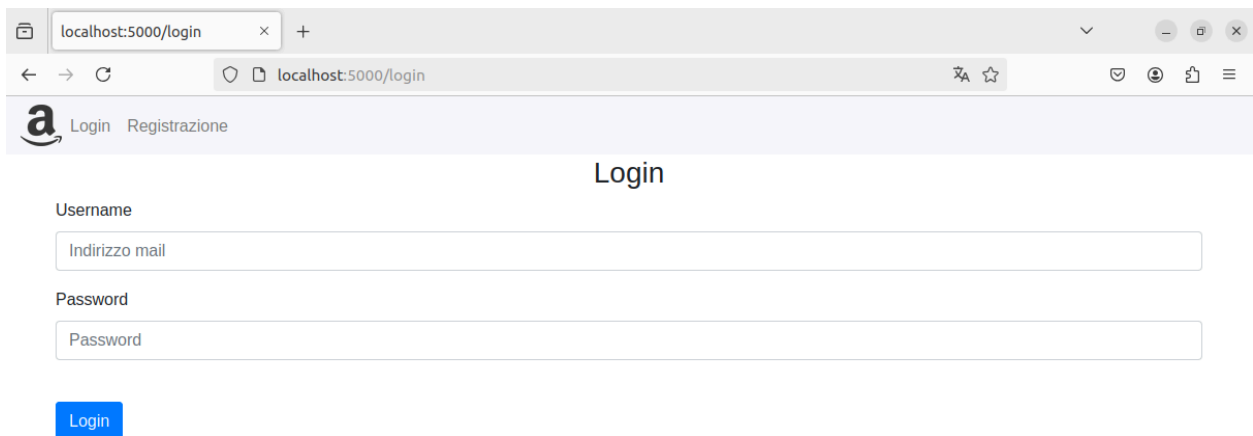
La scelta del progetto Piattaforma di E-commerce ci ha permesso di sviluppare l'applicazione suddividendola opportunamente nelle due parti acquirente e venditore, assegnando un ruolo diverso durante la registrazione dell'utente per poter poi assegnare ad esso le funzionalità corrispondenti.

### 2.1 Esecuzione dell'applicazione

Di seguito una breve spiegazione di come viene eseguita l'applicazione in base ai requisiti suggeriti.

Si avvia andando direttamente alla pagina di login, dove si ha la possibilità di effettuare il login o la registrazione di un nuovo utente.

In Figura 1 la schermata relativa al login di un utente:

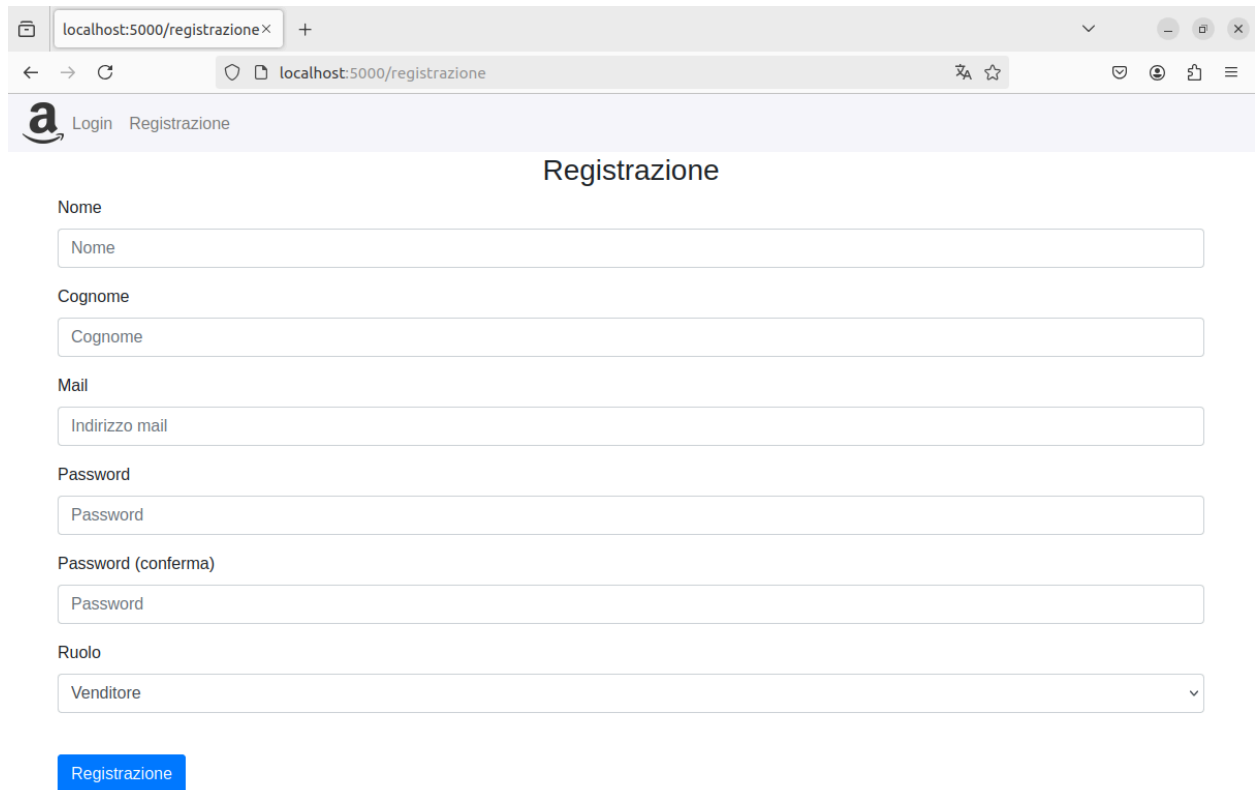


The screenshot shows a web browser window with the address bar displaying 'localhost:5000/login'. The page features a header with an Amazon logo and navigation links for 'Login' and 'Registrazione'. The main section is titled 'Login' and contains two input fields: 'Username' with a placeholder 'Indirizzo mail' and 'Password' with a placeholder 'Password'. Below these fields is a blue 'Login' button.

Figura 1: Login utente

Effettuando la registrazione di un nuovo utente si dovrà scegliere il ruolo per il quale ci si registra, acquirente o venditore, andando poi direttamente alla homepage dedicata in base al ruolo scelto.

In Figura 2 la schermata relativa alla registrazione di un nuovo utente:



The screenshot shows a web browser window with the address bar displaying 'localhost:5000/registrazione'. The page has a header with the Amazon logo and links for 'Login' and 'Registrazione'. The main content area is titled 'Registrazione' and contains a form with the following fields:

- Nome**: A text input field with the placeholder 'Nome'.
- Cognome**: A text input field with the placeholder 'Cognome'.
- Mail**: A text input field with the placeholder 'Indirizzo mail'.
- Password**: A text input field with the placeholder 'Password'.
- Password (conferma)**: A text input field with the placeholder 'Password'.
- Ruolo**: A dropdown menu with 'Venditore' selected.

At the bottom of the form is a blue button labeled 'Registrazione'.

Figura 2: Registrazione utente

Gli acquirenti possono quindi visualizzare i prodotti da acquistare, oppure cercare prodotti da acquistare.

In Figura 3 la schermata relativa all'elenco dei prodotti acquistati:

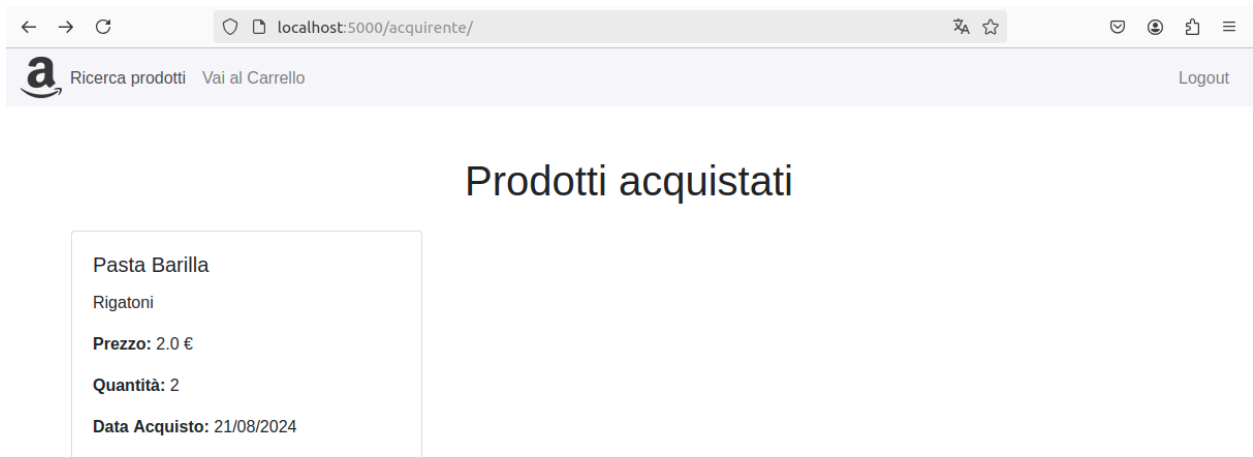
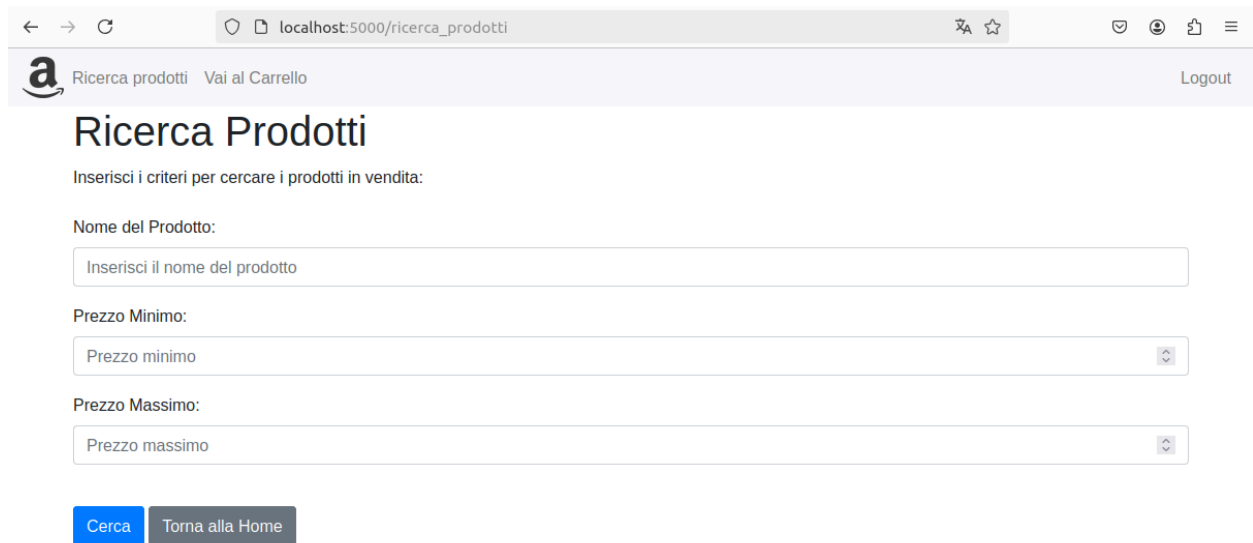


Figura 3: Elenco dei prodotti acquistati

Per cercare un prodotto l'acquirente può selezionare una sottostringa (ad esempio "pasta" oppure "barilla") ma anche un range di prezzo.

In Figura 4 la schermata relativa alla funzionalità di ricerca dei prodotti da acquistare:



The screenshot shows a web browser window with the address bar displaying 'localhost:5000/ricerca\_prodotti'. The page features a header with an Amazon logo, the text 'Ricerca prodotti', a link 'Vai al Carrello', and a 'Logout' button. The main heading is 'Ricerca Prodotti', followed by the instruction 'Inserisci i criteri per cercare i prodotti in vendita:'. Below this, there are three input fields: 'Nome del Prodotto:' with a placeholder 'Inserisci il nome del prodotto', 'Prezzo Minimo:' with a placeholder 'Prezzo minimo', and 'Prezzo Massimo:' with a placeholder 'Prezzo massimo'. At the bottom, there are two buttons: 'Cerca' (highlighted in blue) and 'Torna alla Home'.

Figura 4: Funzionalità di ricerca dei prodotti da acquistare

Inoltre l'acquirente può visualizzare il proprio carrello della spesa, rimuovere o acquistare dei prodotti.

In Figura 5 la schermata relativa al carrello della spesa:

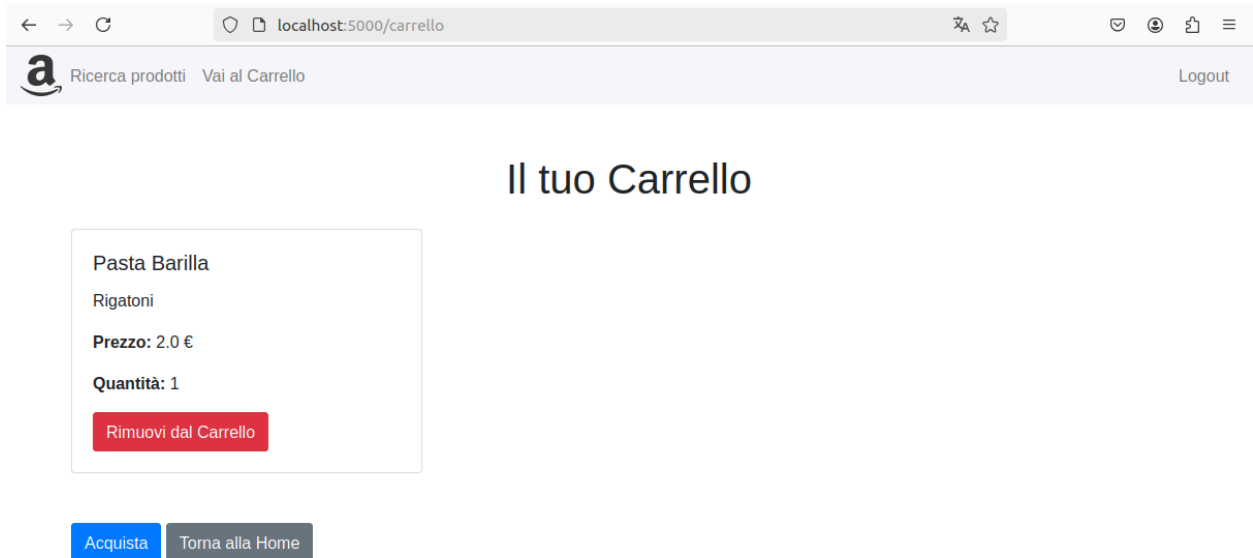


Figura 5: Carrello della spesa



Infine l'acquirente può recensire i prodotti acquistati.

In Figura 6 la schermata relativa alla funzionalità di recensione dei prodotti acquistati:



The screenshot shows a web browser window with the address bar displaying `localhost:5000/recensione/2`. The page header includes the Amazon logo, a search bar with the text "Ricerca prodotti", a link "Vai al Carrello", and a "Logout" button. The main content area features the heading "Sei soddisfatto del tuo acquisto?" followed by the question "Pasta Barilla ha soddisfatto le tue aspettative?". Below this is a prompt "Voto (Per favore inserisci in voto da 1 a 5):" and a text input field. At the bottom of the form are two buttons: "Modifica Recensione" (blue) and "Annulla" (grey).

Figura 6: Funzionalità di recensione dei prodotti acquistati

Per quanto riguarda i venditori invece, questi possono creare nuovi prodotti, modificando o rimuovendo quelli esistenti.

In Figura 7 la schermata relativa alla funzionalità di creazione dei prodotti da vendere

The screenshot shows a web browser window with the address bar displaying 'localhost:5000/vendi\_prodotto'. The page has a header with a logo 'a' and the text 'Metti in Vendita' on the left, and a 'Logout' link on the right. The main heading is 'Metti in vendita il tuo prodotto'. Below this, there are four input fields: 'Nome' with placeholder 'Nome Prodotto', 'Descrizione' with placeholder 'Descrivi il tuo prodotto', 'Prezzo' with placeholder 'Prezzo di vendita' and a dropdown arrow, and 'Quantità' with placeholder 'Quantità di prodotti che desideri vendere' and a dropdown arrow. At the bottom, there are two buttons: 'Conferma' (blue) and 'Torna alla Home' (grey).

Figura 7: Funzionalità di creazione dei prodotti da vendere

Infine il venditore può visualizzare l'elenco dei prodotti in vendita

In Figura 8 la schermata relativa all'elenco dei prodotti in vendita:

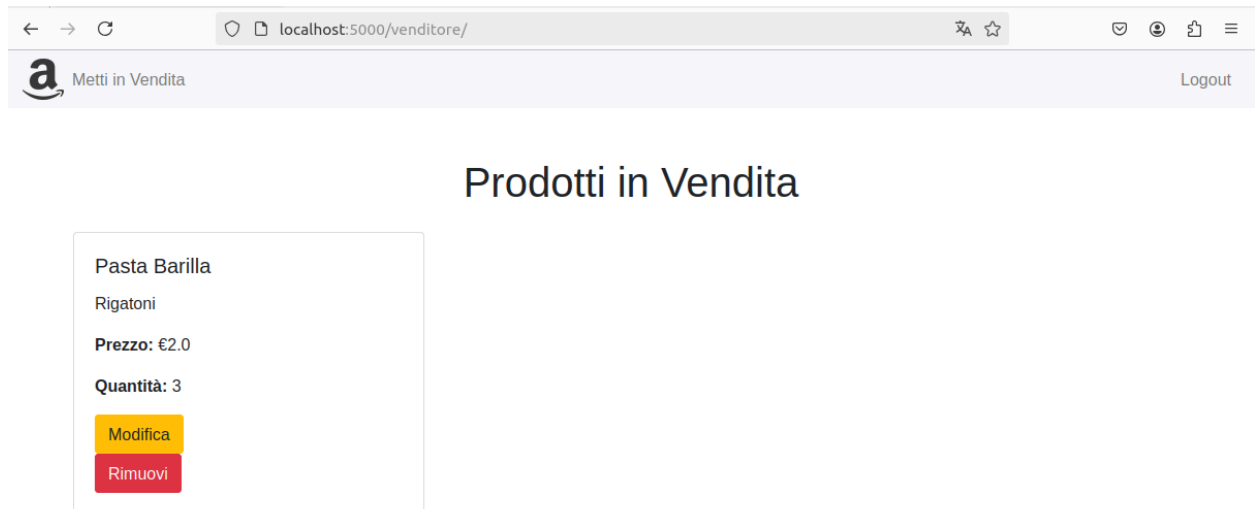


Figura 8: Prodotti in vendita

## **2.2 Requisiti del progetto**

Tutti i requisiti richiesti per lo sviluppo del progetto per la Piattaforma di E-commerce sono stati soddisfatti.

E' stato infatti creato un database per memorizzare le informazioni sui prodotti che i venditori possono gestire.

Gli acquirenti possono cercare i prodotti da acquistare per sottostringa o per intervallo di prezzo.

E' stato implementato un carrello della spesa che permette agli acquirenti di gestire i propri prodotti da acquistare, comprese le disponibilità dei prodotti.

Gli acquirenti possono inoltre visualizzare dati relativi ai propri ordini (come ad esempio la data di acquisto).

infine gli acquirenti possono lasciare delle recensioni riguardanti i prodotti acquistati.

### 3 Progettazione concettuale e logica della basi di dati

In questo paragrafo vengono descritti lo schema grafico ad oggetti, le tabelle, le relazioni ed il modello logico relazionale della base di dati.

#### 3.1 Modello Grafico ad Oggetti

In Figura 9 il modello grafico ad oggetti:

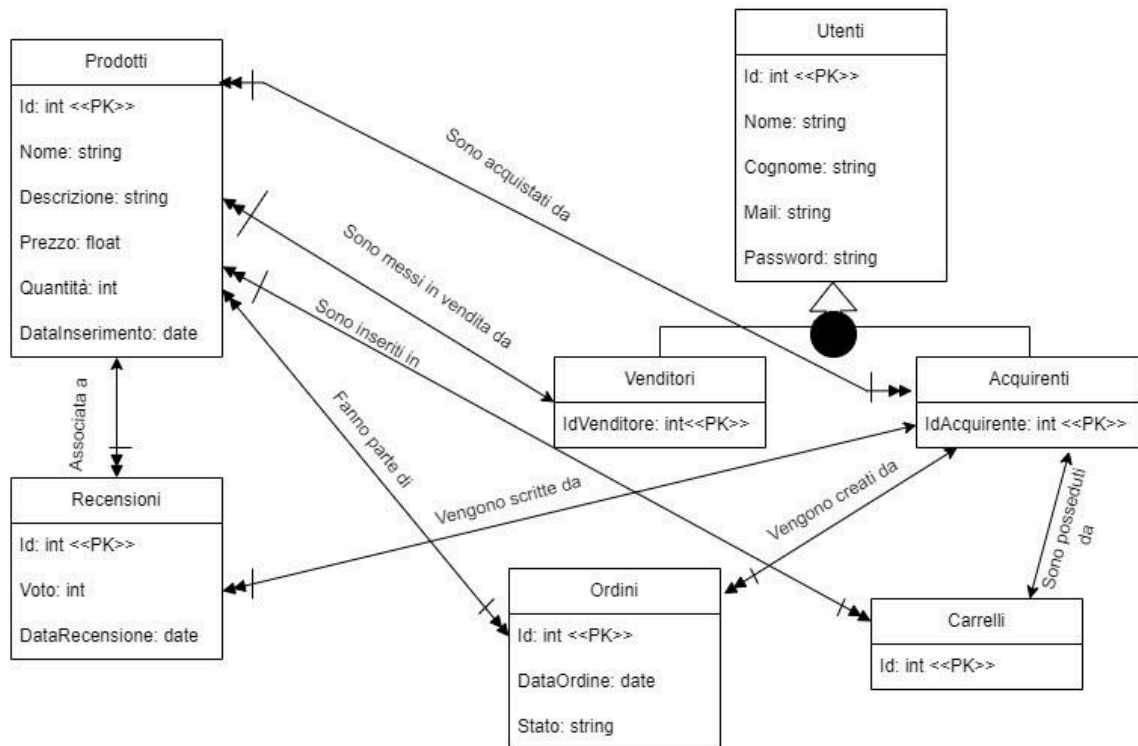


Figura 9: Modello grafico ad oggetti

## 3.2 Tabelle

In questa sezione vengono descritte le tabelle utilizzate per modellare la base di dati.

Le tabelle sono definite come classi ORM (Object Relational Mapping) di SQLAlchemy.

Si utilizzano così funzioni Python per manipolare i dati invece di scrivere direttamente codice SQL.

# Definizione classe (tabella) Utenti

```
class Utenti(db.Model, UserMixin):
```

```
    id = db.Column(db.Integer, primary_key=True, index=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    password = db.Column(db.String(), nullable=False)
    ruolo = db.Column(db.String(10), index=True)
```

# Definizione classe (tabella) Acquirente

```
class Acquirente(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    nome = db.Column(db.String(50), nullable=False)
    cognome = db.Column(db.String(50), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('utenti.id'), index=True)
    user = db.relationship('Utenti', backref='acquirenti')
```

# Definizione classe (tabella) Venditore

```
class Venditore(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    nome = db.Column(db.String(50), nullable=False)
    cognome = db.Column(db.String(50), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('utenti.id'), index=True)
    user = db.relationship('Utenti', backref='venditori')
    prodotti = db.relationship("Prodotto", back_populates="venditore")
```

# Definizione classe (tabella) Prodotto

```
class Prodotto(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    nome = db.Column(db.String, nullable=False)
    descrizione = db.Column(db.String, nullable=False)
    prezzo = db.Column(db.Float, nullable=False)
    quantita = db.Column(db.Integer, nullable=False)
    data_inserimento = db.Column(db.DateTime, default=datetime.now)
    venditore_id = db.Column(db.Integer, db.ForeignKey('venditore.id'), index=True)
    venditore = db.relationship("Venditore", back_populates="prodotti")
    def media_voti(self):
        recensioni = Recensione.query.filter_by(prodotto_id=self.id).all()
        if recensioni:
            return sum(r.voto for r in recensioni) / len(recensioni)
        return None
```

```
class Carrello(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('utenti.id'), nullable=False)
    prodotti = db.relationship('CarrelloProdotto', back_populates='carrello')
```

```
class CarrelloProdotto(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    carrello_id = db.Column(db.Integer, db.ForeignKey('carrello.id'), nullable=False)
    prodotto_id = db.Column(db.Integer, db.ForeignKey('prodotto.id'), nullable=False)
    quantita = db.Column(db.Integer, nullable=False)
    sconto = db.Column(db.Boolean, default=False)
    carrello = db.relationship('Carrello', back_populates='prodotti')
    prodotto = db.relationship('Prodotto')
```

```

class Acquisto(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('utenti.id'), nullable=False)
    prodotto_id = db.Column(db.Integer, db.ForeignKey('prodotto.id'), nullable=False)
    quantita = db.Column(db.Integer, nullable=False)
    data_acquisto = db.Column(db.DateTime, default=datetime.now, nullable=False)
    user = db.relationship('Utenti', backref='acquisti')
    prodotto = db.relationship('Prodotto', backref='acquisti')
    def ha_recensione(self):
        return Recensione.query.filter_by(prodotto_id=self.prodotto_id, user_id=self.user_id).first() is
not None

```

```

class Ordine(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('utenti.id'), nullable=False)
    data_ordine = db.Column(db.DateTime, default=datetime.now)
    prodotti = db.relationship('OrdineProdotto', backref='ordine', lazy=True)
    stato = db.Column(db.String(20), nullable=False, default='acquistato')

```

```

class OrdineProdotto(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    ordine_id = db.Column(db.Integer, db.ForeignKey('ordine.id'), nullable=False)
    prodotto_id = db.Column(db.Integer, db.ForeignKey('prodotto.id'), nullable=False)
    quantita = db.Column(db.Integer, nullable=False)
    spedito = db.Column(db.Boolean, default=False)
    prodotto = db.relationship('Prodotto')

```



```
class Recensione(db.Model):  
    id = db.Column(db.Integer, primary_key=True)  
    prodotto_id = db.Column(db.Integer, db.ForeignKey('prodotto.id'), nullable=False)  
    user_id = db.Column(db.Integer, db.ForeignKey('utenti.id'), nullable=False)  
    voto = db.Column(db.Integer, nullable=False)  
    data_recensione = db.Column(db.DateTime, default=datetime.now, nullable=False)  
  
    prodotto = db.relationship('Prodotto', backref='recensioni')  
    user = db.relationship('Utenti', backref='recensioni')
```

### 3.3 Relazioni

In questa sezione vengono descritte le relazioni scelte per modellare la base di dati:

Prodotti <<— Sono acquistati da —>> Acquirenti

Relazione di tipo M:M che permette ai Prodotti di essere acquistati dagli Acquirenti

Prodotti <<— Sono messi in vendita da —> Venditori

Relazione di tipo M:1 che permette ai Prodotti di essere messi in vendita dai Venditori

Prodotti <<— Sono inseriti in —>> Carrelli

Relazione di tipo M:M che permette ai Prodotti di essere inseriti nei Carrelli

Prodotti <<— Fanno parte di —>> Ordini

Relazione di tipo M:M che permette ai Prodotti di far parte degli Ordini

Recensioni <<— Associate a —> Prodotti

Relazione di tipo M:1 che permette alle Recensioni di essere associate ai Prodotti

Recensioni <<— Vengono scritte da —> Acquirenti

Relazione di tipo M:1 che permette alle Recensioni di essere scritte dagli Acquirenti

Ordini <<— Vengono creati da —> Acquirenti

Relazione di tipo M:1 che permette agli Ordini di essere creati dagli Acquirenti

Carrelli <— Sono posseduti da —> Acquirenti

Relazione di tipo 1:1 che permette ai Carrelli di essere posseduti dagli Acquirenti

### 3.4 Modello Logico Relazionale

In Figura 10 il modello logico relazionale:

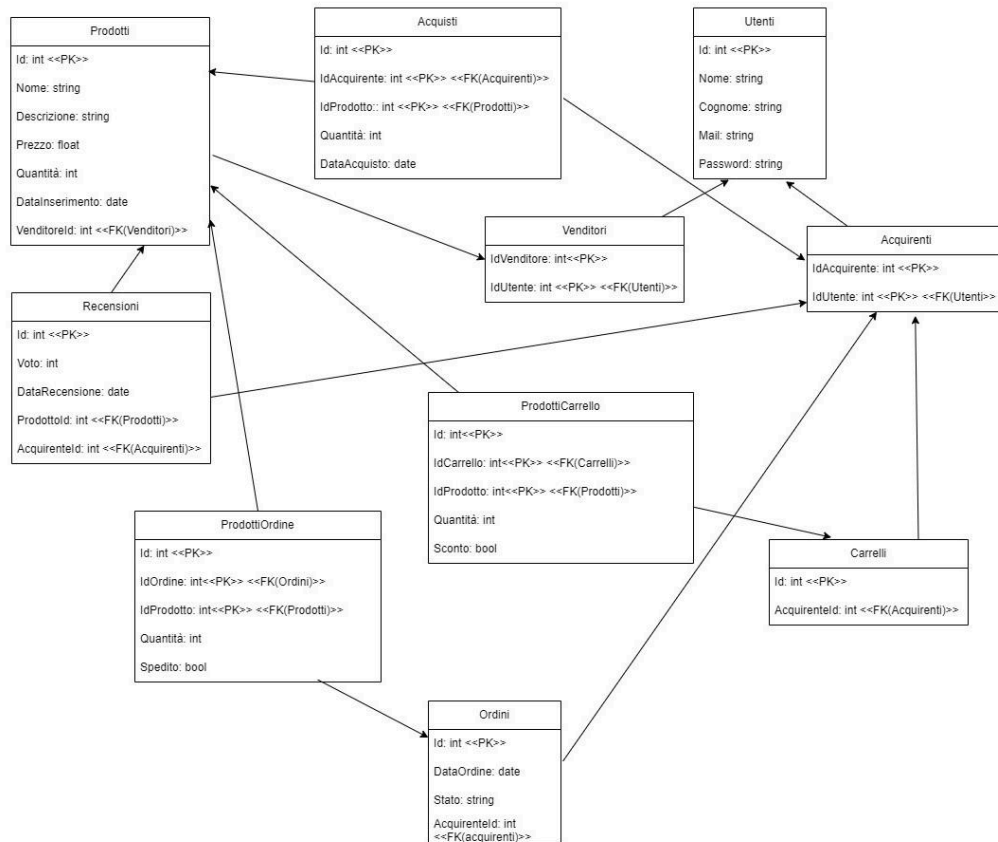


Figura 10: Modello logico relazionale

## 4 Query principali

In questo paragrafo vengono descritte le query principali utilizzate:

```
SELECT * FROM Acquisti WHERE user_id = current_user_id
```

Recupera tutti gli acquisti effettuati dall'utente corrente

```
SELECT * FROM Prodotti WHERE user_id = current_user_id
```

Recupera tutti i Prodotti messi in vendita dall'utente corrente

## 5 Principali scelte progettuali

In questo paragrafo vengono descritte le principali scelte progettuali.

Sono stati innanzitutto creati nel database tre utenti con compiti diversi, amministratore, acquirente e venditore:

```
CREATE USER amministratore WITH PASSWORD 'password';
```

```
CREATE USER acquirente WITH PASSWORD 'password';
```

```
CREATE USER venditore WITH PASSWORD 'password';
```

L'utente amministratore serve per creare gli utenti e fare il login, quindi ha un accesso limitato solo alle tabelle Utente, Acquirente e Venditore.

Abbiamo discusso anche la possibilità di creare come primo utente alla prima esecuzione dell'applicazione, un utente amministratore, il quale poi avrebbe concesso il ruolo venditore su specifica richiesta, mentre tutti gli altri ruoli di default sarebbero stati acquirenti.

Il primo collegamento sarebbe stato indicato da un flag in una apposita tabella di sistema.

Si è optato invece per la scelta del ruolo in fase di registrazione utente visto lo scopo didattico del progetto.

## 5.1 Autorizzazioni

In questa sezione vengono descritte le autorizzazioni che devono essere inserite nel database per poter eseguire l'applicazione:

```
GRANT INSERT, SELECT ON Utente, Acquirente, Venditore TO amministratore;
```

Permette all'utente amministratore di eseguire query di inserimento e lettura sulle tabelle Utente, Acquirente e Venditore.

```
GRANT ALL PRIVILEGES ON SEQUENCE utenti_id_seq TO amministratore;
```

Concede all'utente amministratore tutti i diritti sulla sequenza utenti\_id.

```
GRANT ALL PRIVILEGES ON SEQUENCE acquirente_id_seq TO amministratore;
```

Concede all'utente amministratore tutti i diritti sulla sequenza acquirente\_id.

```
GRANT ALL PRIVILEGES ON SEQUENCE venditore_id_seq TO amministratore;
```

Concede all'utente amministratore tutti i diritti sulla sequenza venditore\_id.

## 5.2 Trigger

In questa sezione vengono descritti i trigger che sono stati utilizzati.

Il primo trigger permette di applicare uno sconto per quantità:

```
CREATE OR REPLACE FUNCTION applica_sconto()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se la quantità è maggiore di 1 e se lo sconto non è già stato applicato
    IF NEW.quantita > 1 AND NOT NEW.sconto THEN
        -- Imposta il flag dello sconto su TRUE
        NEW.sconto := TRUE;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_applica_sconto
BEFORE INSERT OR UPDATE ON carrello_prodotto
FOR EACH ROW
EXECUTE FUNCTION applica_sconto();
```

Il secondo trigger permette di eliminare lo sconto nel caso la quantità di prodotti da acquistare sia tornata all'unità:

```
CREATE OR REPLACE FUNCTION rimuovi_sconto()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se la quantità è diventata minore di 2 per rimuovere lo sconto
    IF NEW.quantita < 2 THEN
        -- Imposta il flag dello sconto su FALSE
        NEW.sconto := FALSE;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trigger_rimuovi_sconto
BEFORE UPDATE ON carrello_prodotto
FOR EACH ROW
WHEN (OLD.quantita >= 2 AND NEW.quantita < 2)
EXECUTE FUNCTION rimuovi_sconto();
```



## 6 Ulteriori informazioni

Per la parte di autenticazione è stata utilizzata la libreria flask\_login.

Per la parte grafica è stato utilizzato il framework Bootstrap.

In tutte le funzioni, per garantire la robustezza del codice, è stato utilizzato il costrutto try-except-finally.

### 6.1 Comandi utili

Di seguito un riassunto dei comandi necessari all'utilizzo dell'applicazione:

Creazione del database

```
# sudo -u postgres psql
postgres=# CREATE DATABASE "progetto";
postgres=# \q
```

Cancellazione del database

```
# sudo -u postgres psql
postgres=# DROP DATABASE "progetto";
postgres=# \q
```

Cambio password di default

```
postgres=# ALTER USER postgres PASSWORD 'Venezia123';
```

Elenco dei database ed accesso

```
# sudo -u postgres psql
postgres=# \l
postgres=# \c progetto
progetto=# \q
```

Riavvio PostgreSQL

```
# service postgresql restart
```

Esecuzione applicazione

```
# export FLASK_APP=main.py
```

```
# flask run
```

In alternativa per l'esecuzione dell'applicazione

```
# python3 main.py
```

Cancellazione tabelle

```
# sudo -u postgres psql
```

```
postgres=# \c progetto
```

```
progetto=# DELETE from utenti;
```

```
progetto=# DELETE from acquirente;
```

```
progetto=# DELETE from venditore;
```

```
progetto=# \q
```

## **7      Contributo al progetto**

Massimo Costantini - 877336

Sviluppo codice (autenticazione)

Documentazione

Samuel Obeng Takyi - 881431

Sviluppo codice (acquirente)

Documentazione

Filippo Tiozzo - 887683

Sviluppo codice (venditore)

Documentazione