E-Commerce

Introduzione

Abbiamo scelto come progetto l'E-Commerce poiché ci sembrava più immediato, di facile comprensione e più diretto.
Come linguaggio per comunicare con Postgres abbiamo utilizzato l' ORM SQLAlchemy poiché risulta il più affidabile tra i tipi di linguaggi utilizzabili direttamente, oltre a non essere limitato da certe debolezze del DBMS.
Inoltre utilizziamo, come richiesto, Flask. Come modulo, risulta estremamente comodo per sviluppare velocemente siti senza dover operare direttamente HTML o CSS.

Funzionalità principali

Avviata l'applicazione, la schermata che ci chiede se ci vogliamo registrare o accedere.
Se selezioniamo la registrazione, dobbiamo aggiungere dati come il proprio nome e cognome, l'email e la password. Inoltre dovremo selezionare se siamo un acquirente o venditore; nel caso di quest' ultimo dovremo anche aggiungere il nome del negozio.

Nel caso entriamo come acquirente avremo le opzioni:

- *Home/Cerca un prodotto*, permette di cercare un prodotto sulla base dei criteri dati all'utente
    - Possiamo accedere alla pagina di un prodotto, dove possiamo visualizzare le sue informazioni ed otteniamo la possibilità di aggiungerlo nel carrello e di dare le recensioni
- *Carrello*, visualizza tutti i prodotti che abbiamo scelto dal sito. Possiamo dare il nostro indirizzo e il modo di pagamento in modo da completare l' ordine
- *I tuoi ordini*, visualizza tutti gli ordini fatti e i loro relativi stati

Nel caso entriamo come venditore, avremo le opzioni:

- *I miei prodotti*, possiamo visualizzare i tipi di prodotti presenti nel database del negozio
- *Aggiungi prodotti*, Dopo aver inserito un dato, puoi visualizzare le recensioni, modificare i dati del prodotto o eliminarlo.
- *Ordini*, visualizza tutte le richieste provenienti dagli acquirenti. Possiamo specificare lo stato corrente dell'ordine

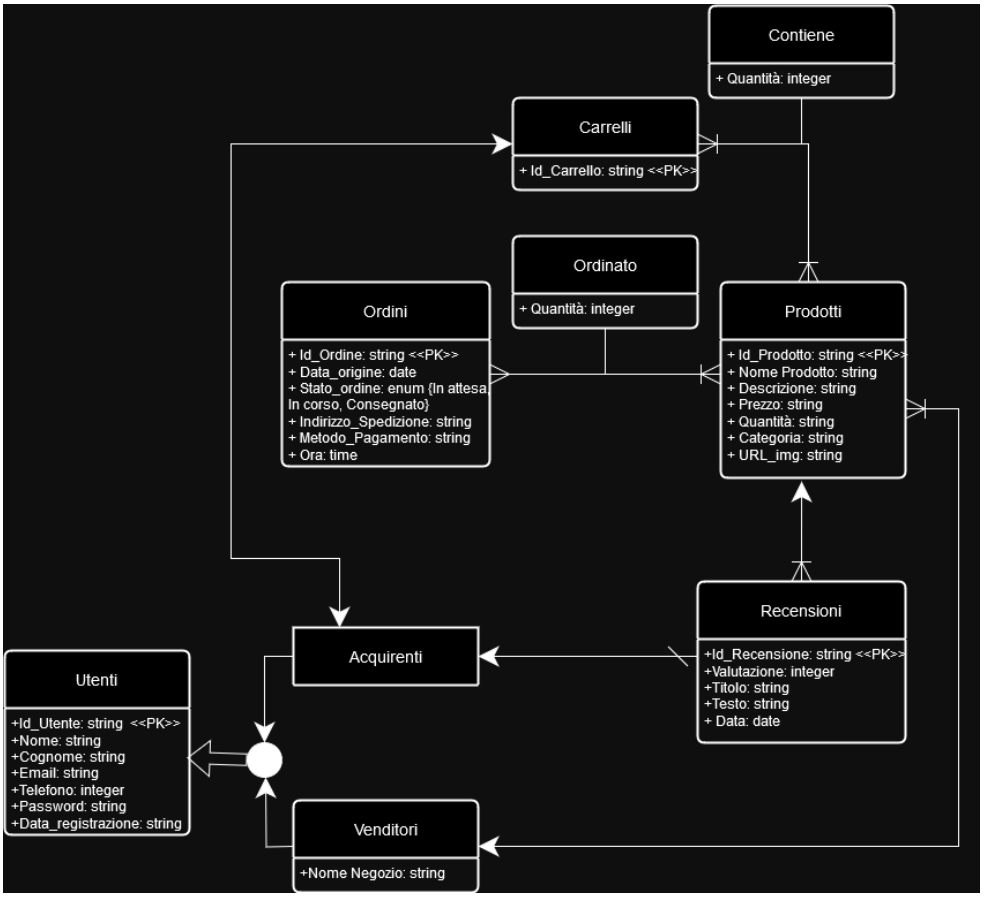
Entrambi hanno

- *Opzioni del profilo*, permette di cambiare la password e altri dati dell' account
- L'opzione di logout

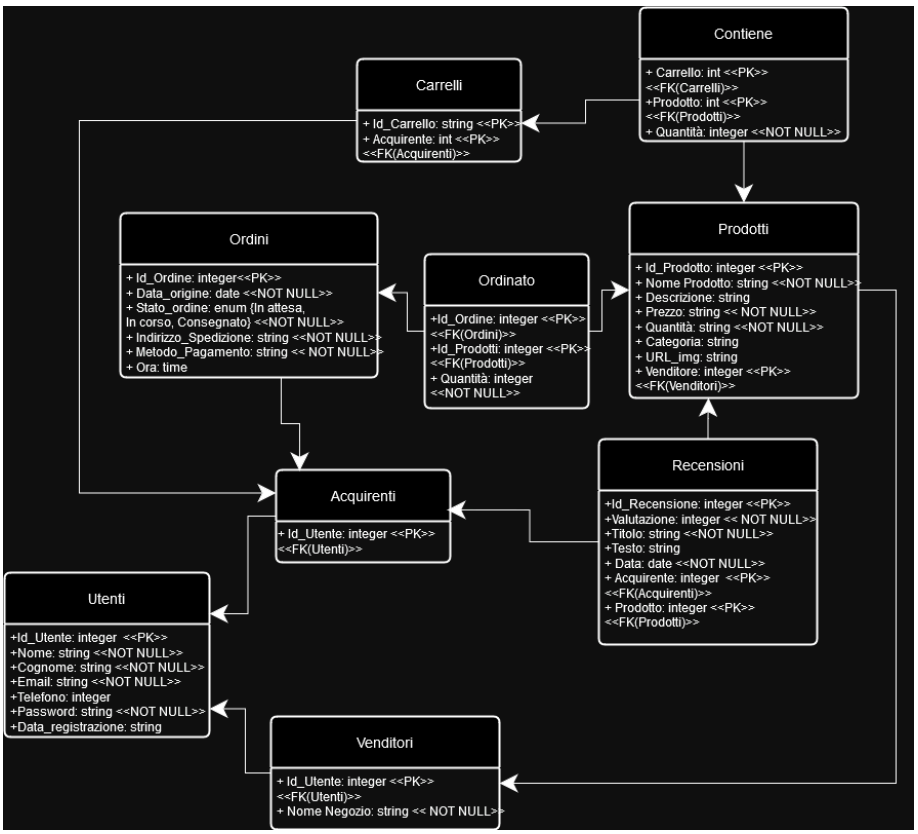
Progettazione concettuale e logica

Acquirenti(Utente <<FK>>)
    Utente(Utenti)
Carrelli(Id_Carrello <<PK>>, Acquirente <<FK>>)
    Acquirente(Acquirenti)
Contiene(Carrello <<FK>>, Prodotto <<FK>>, Quantità)
    Carrello(Carrelli)
    Prodotto(Prodotti)
Ordinato(Ordine <<FK>>, Prodotto <<FK>>, Quantità)
    Ordine(Ordini)
    Prodotto(Prodotti)
Ordini(Id_Ordine <<PK>>, Data_Ordine, Stato_Ordine, Indirizzo_Spedizione, Metodo_Pagamento, Ora, Acquirente <<FK>>)
    Acquirente(Acquirenti)
Prodotti(Id_Prodotto <<PK>>, Nome_Prodotto, Descrizione, Prezzo, Quantità, Cateogoria,URL_Immagine, Venditore <<FK>>)
    Venditore(Venditori)
Recensione(Id_Recensione <<PK>>, Valutazione, Titolo, Testo, Data,Acquirente <<FK>>, Prodotto <<FK>>)
    Acquirente(Acquirenti)
    Prodotto(Prodotti)
Utenti(Id_Utente <<PK>>, Nome, Cognome, Email, Telefono, Password, Data_Registrazione, Indirizzo)
Venditori(Utente <<FK>>, Nome Negozio)
    Utente(Utenti)

Progettazione ad oggetti



Progettazione relazionale



Query principali

- *view_products():* restituisce una lista di tutti i prodotti venduti da un venditore specifico, insieme alla loro valutazione media, se esistono recensioni

```
SELECT Prodotti.*, AVG(Recensioni.Valutazione) AS
media_valutazioni
FROM Prodotti LEFT JOIN Recensioni
ON Recensioni.Prodotto = Prodotti.Id_Prodotto
WHERE Prodotti.Venditore = user_id
GROUP BY Prodotti.Id_Prodotto;
```

- *view_orders()*: Query per ottenere gli ordini contenenti prodotti del venditore corrente

```
SELECT o.id, u.nome, u.cognome
FROM ordini o
JOIN acquirenti a ON o.acquirente = a.id_utente
JOIN utenti u ON a.id_utente = u.id_utente
JOIN ordinato od ON o.id_ordine = od.ordine
JOIN prodotti p ON od.prodotto = p.id_prodotto
JOIN venditori v ON p.venditore = v.utente
WHERE v.utente = $1;   -- Variabile di SQLAlchemy che equivale all'ID utente
```

- purchases(): Recupera tutti gli ordini dell'acquirente

```
SELECT Ordini.*, Ordinato.*, Prodotti.*, Venditori.*
FROM Ordini
JOIN Ordinato ON Ordini.Id_Ordine = Ordinato.Ordine
JOIN Prodotti ON Ordinato.Prodotto = Prodotti.Id_Prodotto
JOIN Venditori ON Prodotti.Venditore = Venditori.Utente
WHERE Ordini.Acquirente = user_id;
```

Scelte progettuali

Per le politiche di integrità, abbiamo:

- Chiavi Primarie: Id_Carrello, Id_Ordine, Id_Prodotto, Id_Recensione, Id_Utente
- Chiavi Esterne: Utente, Acquirente, Prodotto
- NOT NULL: Utenti.Nome, Utenti.Cognome, Utenti.Mail, Utenti.Password, Recensione.Valutazione, Recensione.Titolo, Recensione.Data, Venditore.Nome_Negozio, Ordini.Data_Origine, Ordini.Stato_Ordine, Ordini.Indirizzo_Spedizione, Ordini.Metodo_Pagamento, Ordinato.Quantità, Contiene.Quantità

In quanto alle politiche di integrità, tutte le nostre Chiavi Esterne sono ad eliminazione a cascata.
In quanto ai trigger, ne utilizziamo uno per limitare il numero di ordini sulla base delle disponibilità di magazzino.

```
-- Verifica la quantità disponibile del prodotto
IF (SELECT "Quantità" FROM prodotti WHERE "Id_Prodotto" =
NEW."Prodotto") < NEW."Quantità" THEN
RAISE EXCEPTION 'Quantità richiesta (%s) eccede la disponibilità.',
NEW."Quantità";
END IF;
```

```
-- Aggiorna la quantità disponibile
UPDATE prodotti
SET "Quantità" = "Quantità" - NEW."Quantità"
WHERE "Id_Prodotto" = NEW."Prodotto";
```

Non abbiamo particolari ruoli o indici interni al database.

Scelte tecnologiche specifiche

Come librerie abbiamo utilizzato:

- *Flask* - per rendere automatizzato lo sviluppo della parte HTML
- *SQLAlchemy* - per interagire con il database Postgres
- *Flask_SQLAlchemy* - funziona come layer di comunicazione tra le altre 2 librerie
- *Werkzeug.security* - Viene utilizzato per generare e controllare l'hash di una password

Contributo al progetto

- Francesco Mottin - 894673 - Scrittura codice e documentazione
- Giovanni Mottin - 895357 - Schema ad oggetti e relazionale, scrittura codice
- Zhenyang Zhou - 895909 - Scrittura codice e lato grafico pagine HTML