Basi di Dati e Web

Michele Cairone 284972

Leonardo Minaudo 297792

Descrizione

E-Commerce Macelleria è una piattaforma web per la vendita online di alimenti quali carni e prodotti affini.

Navigando nell'e-commerce è possibile osservare nella home page tutti i prodotti acquistabili, applicare dei filtri di ricerca ad essi e aggiungerli al carrello impostando la quantità desiderata. Cliccando sulla card di un singolo prodotto si può aver modo di visualizzare ulteriori dettagli su di esso.

Un cliente che desidera effettuare un acquisto deve prima autenticarsi con le proprie credenziali (email e password). Qualora non disponesse già di un account può registrarsi cliccando sull'icona del profilo e successivamente sulla voce "Registrati". Atal punto l'utente sarà reindirizzato nella pagina di registrazione dove inserirà tutti i dati utili alla creazione del proprio account per poter effettuare l'accesso.

A login avvenuto il cliente può continuare a effettuare la propria spesa, accedere al proprio carrello, modificare le informazioni personali e visualizzare gli ordini già effettuati.

Il carrello, accessibile dall'icona specifica nella barra di navigazione, include tutti i

prodotti precedentemente aggiunti con i relativi dettagli per ogni articolo e la possibilità di effettuare il check-out tramite pagamento con conto PayPal.

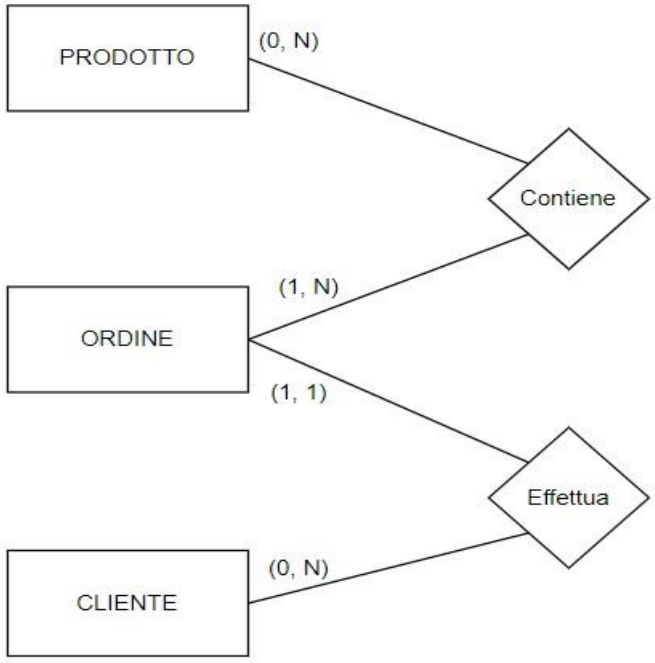
Gli ordini effettuati dai clienti sono fruibili dall'account del commerciante (Admin) che accedendo può visualizzare nel dettaglio i singoli ordini e modificare il loro stato, in modo da tenere aggiornati i propri clienti sulla preparazione del collo. Il commerciante inoltre può aggiungere, rimuovere o modificare i prodotti in vendita nel proprio e-Commerce tramite apposita interfaccia.

Progetto del database

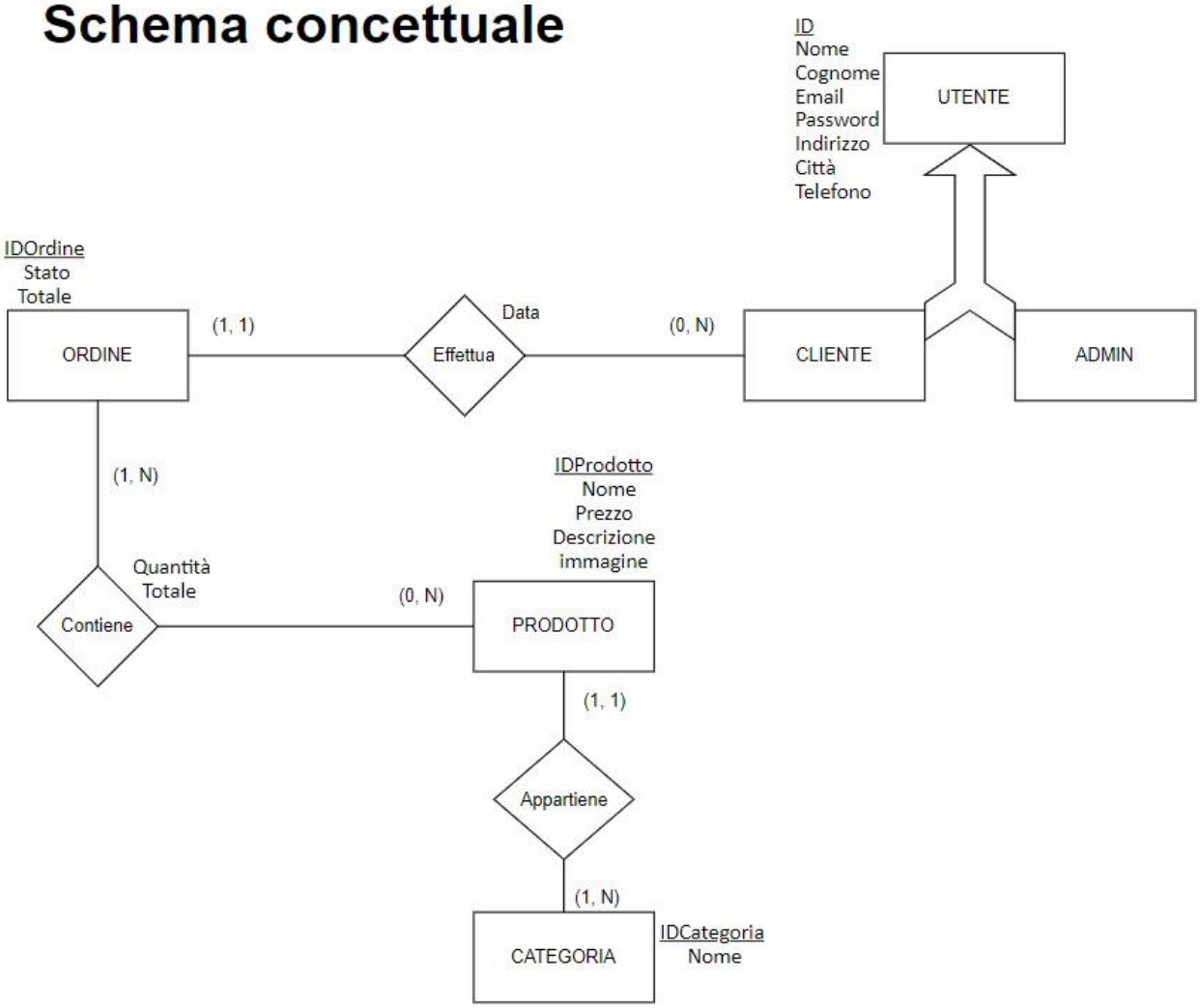
Si è realizzato una base di dati per la consistenza e permanenza dei dati, in cui si vogliono memorizzare i dati degli utenti (nome, cognome, indirizzo, città, telefono, email, password) per poter effettuare gli ordini dei vari prodotti identificati da un codice univoco, nome del prodotto, prezzo al kg, descrizione, immagine e categoria.

Schema scheletro

Entità Relazioni binarie



Schema concettuale



SCHEMA LOGICO

Clients (ID, Name, Surname, Email, Password, Address, City, Telephone, Admin)

- ➔ Vincoli:
  - ID: PRIMARY KEY
  - E-mail: UNIQUE
  - Name, Surname, Email, Password, Address, City, Telephone, Admin: NOT NULL

Category (ID, Name)

- ➔ Vincoli:
  - ID: PRIMARY KEY
  - Name: NOT NULL

Products (ID, Name, Price, Description, Image, Id_category)

- ➔ Vincoli:
  - ID: PRIMARY KEY
  - Name, Price: NOT NULL
  - Products.Id_category → Category.ID

Orders (ID, ID_Clients, Date_ord, State, Total)

- ➔ Vincoli:
  - ID: PRIMARY KEY
  - Orders.ID_Clients → Clients.ID
  - Date_ord, State, Total: NOT NULL

Order_detail (ID_Order, ID_Product, Amount, Total)

- ➔ Vincoli:
  - ID_Order, ID_product: PRIMARY KEY
  - Order_detail.ID_Order → Orders.ID
  - Order_detail.ID_Product→ Products.ID
  - Amount, Total: NOT NULL

Ipotesi sui vincoli e scelte progettuali:

1. Abbiamo aggiunto ad ogni tabella un codice identificativo univoco in modo tale da semplificare e facilitare la gestione dei FOREIGN KEY e PRIMARY KEY.

2. L'e-mail del cliente è stata resa UNIQUE perché non possono esistere due clienti registrati con la stessa e-mail. Tuttavia, non si è voluto usare la mail come chiave primaria per una più facile gestione dei collegamenti con gli ordini tramite il codice univoco del cliente (chiave primaria).

3. Nella tabella CLIENTS si è aggiunto l'attributo ADMIN di tipo booleano per gestire il tipo di login nel sito, se valorizzato a TRUE si accede al sito in modalità amministratore in modo tale da poter accedere alla sessione degli ordini e alla sessione della gestione dei prodotti, altrimenti se valorizzato a FALSE, come di default, si accede in modalità cliente. Questo tipo di decisione progettuale si può identificare come una generalizzazione in cui si sono accorpati i figli nel padre.

4. Per quanto riguarda l'associazione APPARTIENE (uno a molti) non abbiamo voluto accorpare le associazione in un'unica relazione (tabella PRODUCTS), perché volevamo lasciare le due entità separate per eventuali aggiunte di nuove categorie ecc..

5. L'associazione EFFETTUA (uno a molti) è stata accorpata nell'entità ORDERS.

6. Invece, l'associazione CONTIENE (molti a molti) è stata tradotta in una relazione (tabella: ORDER_DETAIL) che mantiene i dettagli di ogni singolo ordine .

Interrogazioni realizzate per il funzionamento del sito:

1. Query usata per selezionare tutti i prodotti presenti nel database:
   SELECT * FROM products;

2. Query usata per selezionare il prodotto con quel specifico ID:
   SELECT * FROM products WHERE id = :id;

3. Query usata per selezionare tutti gli ordini presenti nel database:
   SELECT * FROM orders;

4. Query usata per selezionare l'ordine con quel specifico ID:
   SELECT * FROM orders WHERE id = :id;

5. Query usata per selezionare l'ordine con un ID specifico e recuperare i singoli prodotti presenti nell'ordine:
   SELECT s.id_order, o.date_ord, p.id, p.name, p.image, s.amount, s.total, o.state
   FROM products p, order_detail s, orders o
   WHERE p.id = s.id_product AND s.id_order = :id AND o.id = s.id_order;

6. Query usata per recuperare tutti gli ordini effettuati da un cliente ordinati in modo decrescente rispetto all'ID dell'ordine:
   SELECT o.id AS id_order, o.date_ord, o.state, o.total
   FROM orders o
   WHERE o.id_client = :id
   ORDER BY o.id DESC;

7. Query usata per recuperare tutte le informazioni di uno specifico cliente:
   SELECT * FROM clients WHERE id = :id;

8. Query usata per recuperare tutte le informazioni del cliente e di un suo specifico ordine:
   SELECT c.id as 'id_client', c.name as 'name_client', c.surname, c.address, c.city,
   c.telephone,s.id_order, state, date_ord, p.id, p.name, p.image, s.amount, s.total
   FROM clients c, orders o, products p, order_detail s
   WHERE o.id_client = c.id AND p.id = s.id_product AND o.id = s.id_order AND s.id_order = :id ;

9. Query usata per recuperare la lista con le informazioni degli ordini di tutti i clienti con tutte le informazioni dei clienti, ordinati in maniera decrescente rispetto alla data dell'ordine:
   ```
   SELECT o.id AS 'id_order', c.id, name, surname, state, date_ord, total
   FROM clients c, orders o
   WHERE o.id_client = c.id
   ORDER BY date_ord DESC;
   ```

10. Query usata per inserire nel database i dati del nuovo ordine effettuato da un cliente:
    ```
    INSERT INTO orders(id, id_client, date_ord, state, total)
    VALUES(null, :id_client, :date_ord, :state, :total) ;
    ```

11. Query usata per recuperare l'ID dell'ultimo ordine effettuato da un cliente:
    ```
    SELECT id AS id_order
    FROM orders
    WHERE id_client = :id_client
    AND date_ord = (SELECT MAX(date_ord) FROM orders WHERE id_client = :id_client) ;
    ```

12. Query usata per inserire nel database i dettagli di un nuovo ordine, cioè i prodotti acquistati con quantità e prezzo :
    ```
    INSERT INTO order_detail (id_order, id_product, amount, total)
    VALUES(:id_order, :id_product, :amount, :total) ;
    ```

13. Query usata per aggiungere al database un nuovo prodotto:
    ```
    INSERT INTO products(id, name, price, description, image, id_category)
    VALUES(null, :name, :price, :description, :image, :id_category) ;
    ```

14. Query usata per recuperare i prodotti che appartengono ad una determinata categoria specificando il nome della categoria:
    ```
    SELECT p.id, p.name, p.price, p.description, p.image
    FROM products p, category c
    WHERE p.id_category = c.id AND (c.name = NOME CATEGORIA) ;
    ```

15. Query usata per recuperare i prodotti che appartengono ad una determinata categoria oppure ad un'altra:
    ```
    SELECT p.id, p.name, p.price, p.description, p.image
    FROM products p, category c
    WHERE p.id_category = c.id AND (c.name = 'Carne Bianca') OR (c.name = Carne Rossa) OR (c.name = Preparati) ;
    ```

16. Query usata per recuperare e verificare se l'utente che sta effettuando il login esiste:
    ```
    SELECT * FROM clients
    WHERE email = :email;
    ```

17. Query usata per aggiornare i dati presenti nel database di un determinato cliente:
    ```
    UPDATE clients
    SET name= :name, surname = :surname, address = :address, city= :city, email =:email, password= :password, telephone =:telephone
    WHERE clients.id = :id;
    ```

18. Query usata per aggiornare i dati presenti nel database di un determinato prodotto:
    UPDATE products
    SET name= :name, price = :price, description= :description, image =:image, id_category =:id_category
    WHERE products.id = :id;

19. Query usata per aggiornare lo stato di un determinato Ordine:
    UPDATE orders
    SET state= :state WHERE orders.id = :id;

20. Query usata per eliminare un prodotto dal database specificando il suo ID:
    DELETE FROM products WHERE id = :id;