CANDYSHOP – PROGETTO BASI DI DATI

BELUSSI ALESSANDRO

Sommario

INTRODUZIONE

Il progetto trae ispirazione da un autentico Candyshop che ha fornito lo spunto per sviluppare l'idea su cui si basa. Si vuole realizzare un sito web e-commerce che permetta l'acquisto di prodotti relativi ad una catena di negozi di Candyshop. In particolare, è necessaria la realizzazione di un database in modo tale da memorizzare i prodotti venduti e gli ordini effettuati dai clienti. All'interno del sito web vengono introdotte anche altre funzionalità oltre all'acquisto dei prodotti come la possibilità di registrarsi e avere un proprio profilo. Inoltre, per l'amministratore è possibile caricare dei nuovi prodotti, registrare dei nuovi clienti, visualizzare lo storico degli ordini di tutti i clienti e visualizzare diversi report relativi ai prodotti nei diversi negozi.

La catena di Candyshop possiede diverse sedi. Ogni negozio viene identificato da un apposito codice e si vuole considerare il nome del negozio, il numero di dipendenti, il telefono, l'indirizzo mail e l'orario lavorativo.

In ogni negozio lavorano diversi dipendenti ognuno con un proprio ruolo; ad esempio, vi sono gli store manager, assistant manager, il commesso e il cassiere. Per ogni lavoratore si vogliono memorizzare le proprie informazioni personali quali nome, cognome, codice fiscale, la data di nascita, l'età e altre informazioni come il numero di telefono e lo stipendio. Ogni dipendente ha un contratto e una mansione che ricopre all'interno del negozio in un determinato turno mentre il titolare, del quale si vuole conoscere la data di nomina dell'incarico, si occupa della gestione del negozio. In particolare, può consultare il sito web per visualizzare i prodotti che devono essere riforniti, può visualizzare i diversi ordini effettuati e le recensioni rilasciate dagli utenti.

Ciascun negozio vende diversi prodotti come per esempio barattoli di caramelle, bomboniere, torte di marshmallow, confetti e dragges. Ogni prodotto ha un nome che lo identifica all'interno del catalogo, una descrizione, un peso e un prezzo. Di ogni prodotto si deve conoscere la quantità presente in magazzino e la categoria che permette di raggruppare tra loro i diversi prodotti. La categoria permette di descrivere per un gruppo di prodotti la data di scadenza e la soglia minima di prodotti dopo la quale deve essere effettuato il riassortimento delle scorte presso il fornitore. Ogni categoria viene identificata mediante un codice e racchiude alcune informazioni in comune tra i diversi prodotti come il packaging utilizzato e la data di scadenza dei prodotti. Inoltre, la categoria ha un nome e una descrizione che permette di specificare maggiori informazioni relative ai prodotti.

Ogni prodotto viene acquistato dai fornitori e successivamente venduto dai diversi Candyshop. Il proprietario dell'azienda fornitrice stabilisce un contratto con il titolare del negozio all'interno del quale vengono definiti i prodotti e le relative quantità che vengono fornite settimanalmente. Ogni azienda ha un codice che la identifica e un nome; inoltre, si vuole considerare anche un recapito come il numero di telefono e la sede.

I clienti possono accedere al sito per visualizzare i prodotti disponibili, per effettuare degli acquisiti e per visualizzare lo storico degli ordini. Gli utenti possono registrarsi al sito fornendo un nome utente, un'e-mail e una password e successivamente possono accedere mediante lo username e la rispettiva password. Gli utenti possono effettuare degli ordini online selezionando i diversi prodotti e fornendo le informazioni di spedizione come l'indirizzo di consegna. Per ogni ordine si vuole memorizzare la data dell'ordine, la data di consegna e in base alla spesa vengono generati dei punti fedeltà che permettono di classificare l'utente in diversi livelli fedeltà come standard, premium e gold.

Il livello di fedeltà viene definito attribuendo un punto ogni due euro di spesa e considerando i seguenti range:

- standard: 0 – 499;
- premium: 500 – 1499;
- gold: superiore a 1500.

A seconda del livello, ogni utente ha a disposizione delle speciali promozioni e degli sconti periodici. Infine, per ogni acquisto si vuole tener traccia anche della modalità di spedizione che può essere standard, express oppure ritiro in negozio. Il cliente può lasciare delle recensioni tramite un punteggio, un commento e si vuole memorizzare anche la data in cui è stata rilasciata.

PROGETTAZIONE CONCETTUALE

Si procede con la realizzazione del database e in particolare con la progettazione concettuale all'interno della quale viene definito lo schema E/R. Dopo aver analizzato i requisiti e distinto i dati dalle funzionalità che devono essere offerte, si individuano le seguenti entità fondamentali:

- SEDE: rappresenta l'ubicazione cioè il luogo fisico in cui si trovano i diversi negozi e i fornitori.
- NEGOZIO: rappresenta il negozio, il punto vendita della catena di Candyshop.
- ORARIO: rappresenta i diversi orari di apertura e di chiusura settimanali dei negozi.
- LAVORATORE: rappresenta i diversi lavoratori all'interno dei negozi ed è un'entità padre della seguente generalizzazione parziale esclusiva in quanto ci potrebbero essere altri lavoratori che non sono né il titolare né il dipendente e si considera che un dipendente non sia un titolare.
    - TITOLARE: rappresenta il titolare del negozio ed è l'entità figlia della generalizzazione con l'entità padre lavoratore.
    - DIPENDENTE: rappresenta i diversi commessi all'interno del negozio ed è l'entità figlia della generalizzazione con l'entità padre lavoratore.
- TURNO: rappresenta l'orario lavorativo del lavoratore.
- PRODOTTO: rappresenta i diversi prodotti venduti nei negozi.
- CATEGORIA: rappresenta le diverse categoria in cui sono raggruppati i prodotti.
- FORNITORE: rappresenta coloro che vendono i diversi prodotti ai negozi.
- UTENTE: rappresenta coloro che possono accedere al sito ed è l'entità padre della seguente generalizzazione parziale esclusiva perché vi potrebbero essere altri utenti come il dipendente del negozio ed esclusiva perché l'amministratore non è un cliente.
    - CLIENTE: rappresenta coloro che effettuano l'acquisto fisicamente in negozio oppure online.
    - AMMINISTRATORE: rappresenta colui che gestisce il sito web.
- SPESA: rappresenta l'ordine effettuato dal cliente ed è l'entità padre della seguente generalizzazione is-a in quanto l'ordine online è una spesa.
- ORDINE ONLINE: rappresenta l'ordine che viene effettuato direttamente dal sito web.
- RECENSIONE: rappresenta la valutazione rilasciata dagli utenti in merito agli acquisti effettuati.

In particolare, viene realizzata un'entità sede per rappresentare il luogo fisico sia dell'entità negozio che dell'entità fornitore; in alternativa, si sarebbero potuti considerare degli attributi nelle due entità che rappresentassero la sede.

Inoltre, si considera un attributo mansione per rappresentare i diversi ruoli ricoperti dai dipendenti del negozio in quanto non si hanno ulteriori informazioni in merito per poter realizzare una generalizzazione e quindi considerare ulteriori entità.

Stabilite le entità coinvolte e i rispettivi attributi, vengono indicati gli identificatori cioè le chiavi che permettono di individuare in modo univoco le istanze dell'entità.
Le scelte vengono motivate di seguito:

- SEDE: la chiave è data dall'attributo città e indirizzo; infatti, l'attributo città non è sufficiente a identificare la sede in quanto si potrebbero avere più sedi nella stessa città mentre l'attributo indirizzo non è sufficiente perché il CAP non è univoco per ogni paese.
- NEGOZIO: la chiave è data dall'attributo codiceNegozio.
- ORARIO: la chiave è data da un identificatore esterno composto dall'attributo giorno e dalla chiave dell'entità NEGOZIO ovvero codiceNegozio; pertanto, si tratta di un'entità debole. Siccome si prende in considerazione la possibilità che l'orario possa appartenere anche ad altri negozi, il solo attributo giorno non sarebbe sufficiente a identificare l'istanza e quindi si considera anche l'attributo codiceNegozio.
- LAVORATORE: la chiave è data dall'attributo codiceFiscale.
- TURNO: la chiave è data dall'attributo codiceTurno che permette di identificare ciascun turno.
- PRODOTTO: la chiave è data dall'attributo nome che si considera univoco per ciascun prodotto del negozio.
- CATEGORIA: la chiave è data dall'attributo codiceCategoria.
- FORNITORE: la chiave è data dall'attributo codiceFornitore.
- UTENTE: la chiave è data dall'attributo username che permette di identificare in modo univoco i diversi utenti.
- ORDINE: la chiave è data dall'attributo codiceOrdine.
- RECENSIONE: la chiave è data dall'attributo numeroProgressivo.

Successivamente si decide la cardinalità per ciascuna relazione che vengono motivate di seguito facendo riferimento al nome dell'associazione:

- TROVA cardinalità uno a uno: ogni negozio è associato ad una e una sola sede mentre una sede può avere un solo negozio; si considera che ogni negozio abbia una sede e che in quella determinata sede vi sia un unico negozio.
- APERTURA cardinalità molti a molti: ogni negozio ha almeno un orario di apertura ma potrebbe averne anche molti considerando orari diversi per i giorni della settimana; ogni orario è relativo ad almeno un negozio.
- IMPIEGO cardinalità molti a molti: ogni negozio ha almeno un lavoratore impiegato presso il proprio negozio mentre il lavoratore può essere impiegato in almeno un negozio. Sull'associazione si considerano anche due attributi dell'associazione ovvero dataInizio e dataFine che permettono di prendere in considerazione anche coloro che non lavorano più presso il negozio. In particolare, l'attributo d'associazione dataFine è opzionale.
- EFFETTUA cardinalità uno a molti: ogni lavoratore può effettuare un solo turno contemporaneamente mentre un turno può essere effettuato da più lavoratori.
- APPARTIENE cardinalità uno a molti: ogni prodotto appartiene a una sola categoria mentre ad una categoria possono appartenere più prodotti.
- VENDE cardinalità molti a molti: ogni negozio vende almeno un prodotto mentre lo stesso prodotto può essere venduto da più negozi (si intende la stessa tipologia di prodotto).

- CONSEGNA cardinalità uno a molti: ogni prodotto viene realizzato da un fornitore il quale può realizzare più prodotti.
- LOCALIZZATO cardinalità uno a uno: ogni fornitore ha un'unica sede mentre ad una sede appartiene un unico fornitore.
- COMPIE cardinalità molti a uno: ogni utente può realizzare almeno una spesa mentre la spesa può essere realizzata da un unico utente.
- ASSOCIATA cardinalità uno a molti: ogni spesa è associata ad un solo negozio nel quale possono essere effettuate più spese.
- ORDINA cardinalità molti a molti: ogni spesa ordina/contiene almeno un prodotto il quale può essere ordinato da più spese.
- RICOPRE cardinalità uno a uno: ogni amministratore può ricoprire il ruolo di titolare il quale può essere un amministratore.
- RILASCIA cardinalità molti a uno: ogni utente può rilasciare zero o più recensioni mentre una recensione viene rilasciata da un unico utente.

Di seguito viene riportato lo schema E/R:

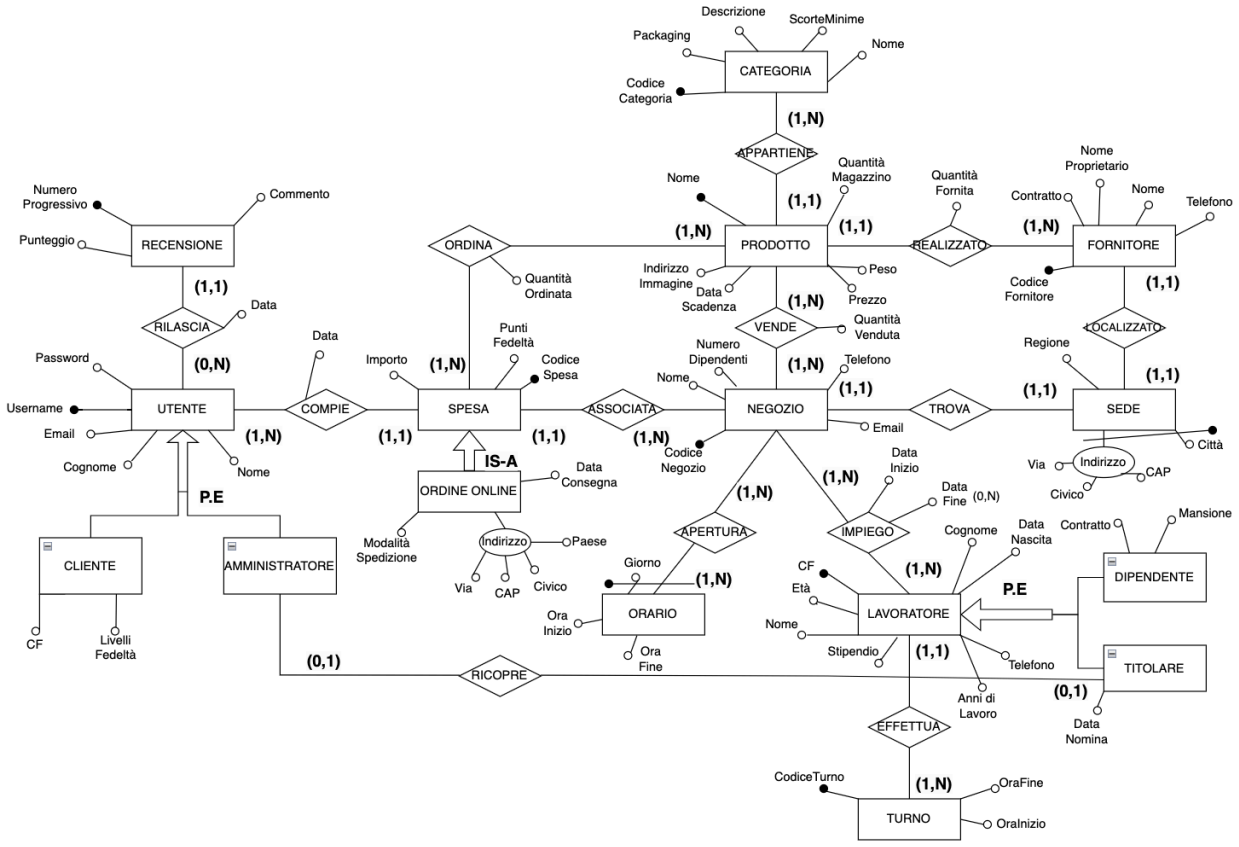


Figura 1: Schema E/R

PROGETTAZIONE LOGICA

Dopo aver realizzato lo schema E/R durante la progettazione concettuale si procede con la progettazione logica all'interno della quale viene realizzato il modello logico dei dati. Poiché si intende realizzare un database relazionale, il modello logico adottato sarà un modello relazione in cui i dati vengono organizzati in relazioni.
La progettazione logica si compone di due fasi:

- la ristrutturazione dello schema E/R durante la quale lo schema E/R viene adattato per eliminare costrutti difficilmente traducibili nel modello relazionale;
- la traduzione in cui si passa dallo schema E/R ristrutturato al modello relazionale.

In particolare, durante la fase di ristrutturazione è necessario eliminare le generalizzazioni attraverso il collasso verso l'alto, il collasso verso il basso o trasformandole in associazioni. Inoltre, eventuali attributi composti devono essere convertiti in attributi semplici mentre eventuali identificatori esterni vengono modificati portando l'attributo chiave esterno sull'entità debole. Infine, viene effettuata anche un'analisi degli attributi e delle relazioni in modo tale da eliminare eventuali ridondanze.

Quindi, considerando lo schema E/R vengono apportate le seguenti modifiche:

- valutazione di attributi e relazioni ridondanti:
    a. il numero di dipendenti per ogni *NEGOZIO* è ridondante in quanto può essere calcolato dall'entità *LAVORATORE*.
    b. l'attributo età dell'entità *LAVORATORE* è ridondante in quanto può essere calcolato a partire dall'attributo dataNascita della stessa entità.
    c. l'attributo anniLavoro dell'entità *LAVORATORE* è ridondante in quanto può essere calcolato dagli attributi dell'associazione *IMPIEGO*.
    d. l'attributo mansione dell'entità *DIPENDENTE*, dopo la ristrutturazione della generalizzazione risulta ridondante in quanto viene introdotto un attributo ruolo per considerare le entità figlie. Quest'ultimo include implicitamente anche l'informazione relativa alla mansione.
- ristrutturazione delle generalizzazioni:
    a. la generalizzazione dell'entità *LAVORATORE* viene ristrutturata con un collasso verso l'alto prendendo in considerazione che ogni attributo dell'entità figlie ovvero di *DIPENDENTE* e di *TITOLARE* diventa un attributo opzionale. Si aggiunge anche un attributo di tipo definito come ruolo per considerare le diverse entità figlie della generalizzazione.
    b. la generalizzazione dell'entità *SPESA* viene ristrutturata considerando un collasso verso l'alto e introducendo un attributo tipo; gli attributi dell'entità *ORDINE ONLINE* diventano opzionali.
    c. la generalizzazione dell'entità *UTENTE* viene ristrutturata considerando un collasso verso l'alto con l'introduzione di un attributo ruolo per tenere traccia delle informazioni delle entità figlie e gli attributi dell'entità figlia *CLIENTE* diventano opzionali. L'associazione dell'entità figlia *AMMINISTRATORE* viene collegata all'entità padre.
- ristrutturazione degli attributi composti:
    a. l'attributo composto indirizzo dell'entità *SEDE* viene gestito considerando attributi semplici; in particolare l'attributo civico viene considerato all'interno dell'attributo via. Il CAP non è necessario nella chiave perché non identifica in modo univoco un paese in quanto vi possono essere più paesi con lo stesso CAP.

b. l'attributo composto indirizzoConsegna dell'entità *ORDINE ONLINE* viene ristrutturato in attributi semplici.

- gestione degli identificatori esterni:
  a. L'identificatore esterno dell'entità *ORARIO* viene ristrutturato introducendo anche la chiave dell'entità *NEGOZIO* e ottenendo una chiave composta dagli attributi giorno e codiceNegozio.

Di seguito viene riportato lo schema E/R ristrutturato:

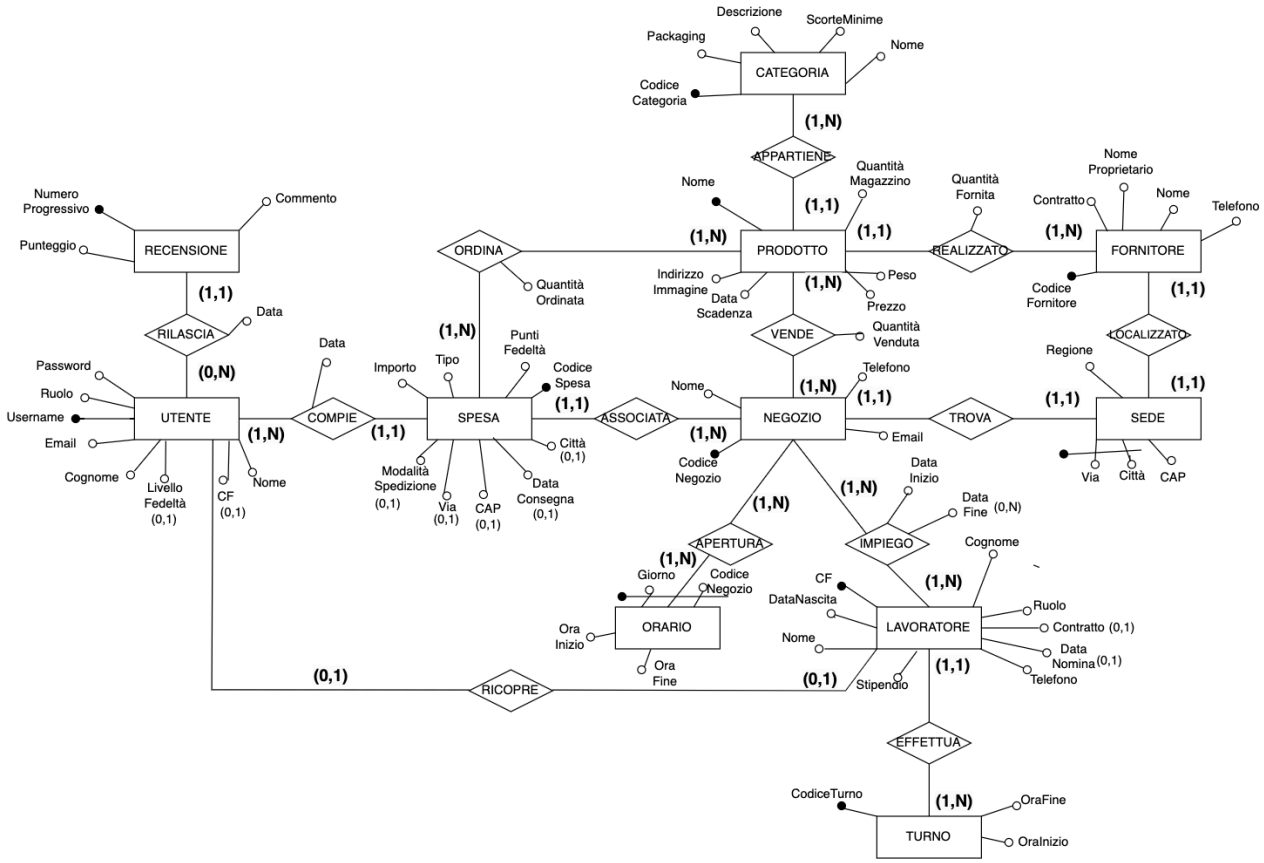


Figura 2: Schema E/R ristrutturato

Procedendo lo schema E/R appena ristrutturato viene tradotto nel modello relazionale considerando le seguenti regole:

- ogni entità viene tradotta in una relazione con lo stesso nome e contenente i suoi attributi;
- associazioni con cardinalità molti a molti vengono tradotte in una relazione in cui vi sono le chiavi dell'entità ed eventuali attributi dell'associazione;
- associazioni con cardinalità uno a molti vengono tradotte introducendo la chiave dell'entità con cardinalità molti ed eventuali attributi d'associazione nell'entità con cardinalità uno; in alternativa, si può considerare una regola simile a quella precedente ovvero si realizza una relazione anche per l'associazione in cui vengono poste le chiavi dell'entità coinvolte ed eventuali attributi d'associazione.
- associazioni con cardinalità uno a uno vengono tradotte introducendo la chiave dell'entità con cardinalità uno ed eventuali attributi d'associazione nell'entità con cardinalità uno.

DI seguito vengono presentate le diverse relazioni dove l'attributo sottolineato rappresenta la chiave primaria mentre l'attributo anticipato dall'asterisco rappresenta la chiave esterna:

- NEGOZIO (CodiceNegozio, Nome, Telefono, E-mail, *Sede)
- ORARIO (Giorno, CodiceNegozio, OraInizio, OraFine)
- APERTURA (*Negozio, *Orario)
- LAVORATORE (CodiceFiscale, Nome, Cognome, DataNascita, Stipendio, Contratto, Telefono, Ruolo, DataNomina, *Turno, *Utente)
- IMPIEGO (*Negozio, *Personale, DataInizio, DataFine)
- TURNO (CodiceTurno, OraInizio, OraFine)
- SEDE (Città, Via, CAP, Regione)
- PRODOTTO (Nome, Prezzo, Peso, QuantitaMagazzino, DataScandenza, PercosoImmagine, *Categoria)
- CATEGORIA (CodiceCategoria, Nome, Descrizione, ScorteMinime, Packaging)
- FORNITORE (CodiceFornitore, Nome, NomeProprietario, Contratto, Telefono, *Sede)
- CONSEGNA (*Fornitore, *Prodotto, QuantitaFornita)
- SPESA (CodiceOrdine, Importo, Città, CAP, Via, Tipo, ModalitaSpedizione, PuntiFedelta, DataConsegna, DataOrdine, *Utente, *Negozio)
- ORDINA (*Spesa, *Prodotto, QuantitaOrdinata)
- UTENTE (Username, Nome, Cognome, Password, E-mail, CodicaFiscale, LivelloFedelta, Ruolo)
- RECENSIONE (NumeroRecensione, Commento, Punteggio, DataRecensione, *Utente)
- VENDITA *(CodiceNegozio, *NomeProdotto, QuantitaVenduta)

PROGETTAZIONE FISICA

Nella progettazione fisica viene realizzato il database mediante l'utilizzo del linguaggio SQL ma vengono anche utilizzati i seguenti strumenti:

- MAMP: è un pacchetto software che permette di utilizzare un ambiente di sviluppo web locale supportando MySQL e il linguaggio PHP;
- phpMyAdmin: è un'applicazione che permette di gestire facilmente i database MySQL e MariaDB mediante un'interfaccia grafica;
- MySQL: è un DBMS cioè un sistema software che permette di gestire i database.
- MySQLWorkbench: è uno strumento che facilita la gestione, la progettazione di database MySQL.

In particolare, l'utilizzo di MySQLWorkbench non era strettamente necessario ma si è scelto di realizzare l'intero database al suo interno sfruttando l'usabilità che questo software offre rispetto a phpMyAdmin. Lo strumento è stato utilizzato principalmente per testare la realizzazione del database che successivamente è stato importato in phpMyAdmin.
Inoltre, alcune tabelle sono state realizzate mediante il supporto del linguaggio PHP che permette di interagire con un database SQL e di inviare codice SQL mentre altre tabelle sono state caricate direttamente in phpMyAdmin sempre utilizzando il codice SQL. Il linguaggio PHP viene utilizzato per interagire con il database ed eseguire operazioni come la creazione delle tabelle, l'inserimento dei dati e l'esecuzione di query. In particolare, è stata utilizzata l'interfaccia ad oggetti PDO di PHP per accedere al database. Di seguito vengono presentate alcune porzioni di codice SQL riguardanti la creazione di tabelle e altre operazioni utili che sono state realizzate.

Ovviamente all'interno del codice sono presenti altre istruzioni per la realizzazione di tutte le tabelle mentre alcuni inserimenti sono stati effettuati direttamente da phpMyAdmin; di seguito sono proposti alcuni esempi.

La creazione della tabella *SPESA* viene effettuata come di seguito:

```
CREATE TABLE IF NOT EXISTS Spesa (
    CodiceOrdine VARCHAR(255) NOT NULL,
    Importo FLOAT NOT NULL,
    CittaConsegna VARCHAR(20),
    CAP VARCHAR(20),
    Via VARCHAR(20),
    Tipo VARCHAR(30) NOT NULL,
    ModalitaSpedizione VARCHAR(30),
    PuntiFedelta INT NOT NULL,
    DataConsegna DATE,
    DataOrdine DATE NOT NULL,
    Username VARCHAR(20) NOT NULL,
    CodiceNegozio VARCHAR(20) NOT NULL,

    PRIMARY KEY (CodiceOrdine),
    FOREIGN KEY (Username) REFERENCES Utente(Username),
    FOREIGN KEY (CodiceNegozio) REFERENCES Negozio(CodiceNegozio)
);
```

Figura 3: Istruzione CREATE TABLE

Mediante l'utilizzo dell'istruzione *CREATE TABLE IF NOT EXISTS* viene creata una tabella se questa non è già presente all'interno del database; al suo interno vengono definiti i campi della tabella con i rispettivi tipi oltre alla chiave primaria introdotta da *PRIMARY KEY* e la chiave esterna *FOREIGN KEY*.

L'inserimento dei dati nella tabella *SPESA* viene effettuata come di seguito:

```
INSERT INTO Spesa
(CodiceOrdine, Importo, CittaConsegna, CAP, Via, Tipo, ModalitaSpedizione,
PuntiFedelta, DataConsegna, DataOrdine, Username, CodiceNegozio)
VALUES (:codice, :importo, :citta, :cap, :via, :tipo,
:spedizione, :punti, :dataConsegna, :dataOrdine, :username, :codiceNegozio)
```

Figura 4: Istruzione INSERT INTO

L'istruzione *INSERT INTO* permette di introdurre dei dati all'interno della tabella dopo aver definito i diversi campi. All'interno della clausola *VALUES* si introducono i valori mentre in questo caso sono definiti dei segnaposto che successivamente vengono sostituiti dai valori reali grazie al binding dei parametri effettuato dal metodo PHP *bindParam()*. L'utilizzo dei segnaposto permette di evitare il verificarsi di SQL Injection ovvero una tecnica di attacco in cui un malintenzionato inserisce codice SQL dannoso attraverso i dati che vengono inviati ad un'applicazione.

Durante l'acquisto l'utente potrebbe selezionare lo stesso prodotto più volte e quindi il campo quantitaVenduta della tabella *VENDITA* deve essere aggiornato opportunamente.

```
INSERT INTO Vendita
(CodiceNegozio, Nome, QuantitaVenduta)
VALUES (:codiceNegozio, :nome, :quantitaOrdinata)
ON DUPLICATE KEY UPDATE QuantitaVenduta = QuantitaVenduta + :quantitaOrdinata
```

Figura 5: Istruzione INSERT INTO con ON DUPLICATE KEY UPDATE

In aggiunta a quanto detto precedentemente, si considera l'uso della clausola *ON DUPLICATE KEY UPDATE* che permette di specificare un aggiornamento specifico quando la chiave primaria è già presente nella tabella. In questo caso, alla quantità venduta viene aggiunta la quantità ordinata.

La gestione degli ordini è stata realizzata come segue: si prende in considerazione la creazione di un ordine fittizio in modo da poter realizzare la tabella *ORDINA* che rappresenta il carrello cioè i prodotti che sono stati ordinati. Successivamente, la tabella *SPESA* viene aggiornata al termine del checkout con le vere informazioni relative all'ordine che è stato effettuato.

```
UPDATE Spesa
SET Importo = :importo, CittaConsegna = :citta, CAP = :cap, Via = :via, ModalitaSpedizione = :spedizione,
    PuntiFedelta = :punti, DataConsegna = :dataConsegna, DataOrdine = :dataOrdine, CodiceNegozio = :codiceNegozio
WHERE CodiceOrdine = :codiceOrdine AND Username = :username
```

Figura 6: Istruzione UPDATE

L'istruzione *UPDATE* permette di aggiornare i dati contenuti nella tabella; anche in questo caso, si utilizzano dei segnaposto per indicare i dati che vengono aggiornati all'interno della clausola *SET* mentre con la clausola *WHERE* si pone la condizione che l'ordine da modificare è quello in cui il codiceOrdine e lo username corrispondono.

Infine, è stata introdotta la possibilità di cancellare tutti i prodotti che sono stati inseriti all'interno del carrello. Per farlo, si devono eliminare le relative righe sia nella tabella *ORDINA* che nella tabella *SPESA*; di seguito viene riportato l'esempio per la tabella *SPESA.*

```
DELETE FROM Spesa
WHERE CodiceOrdine = :codiceOrdine
```

Figura 7: Istruzione DELETE

L'istruzione *DELETE* permette di cancellare una riga dalla tabella che soddisfa la condizione imposta dalla clausola *WHERE*. Una volta che l'utente decide di annullare il proprio ordine e quindi di liberare il carrello, le quantità dei prodotti selezionati vengono ripristinate come prima della selezione.

INTERROGAZIONI AL DATABASE

All'interno del sito web è presenta una pagina di report, accessibile solo dall'amministratore, che presenta i risultati di diverse query di utilità per la gestione del negozio.

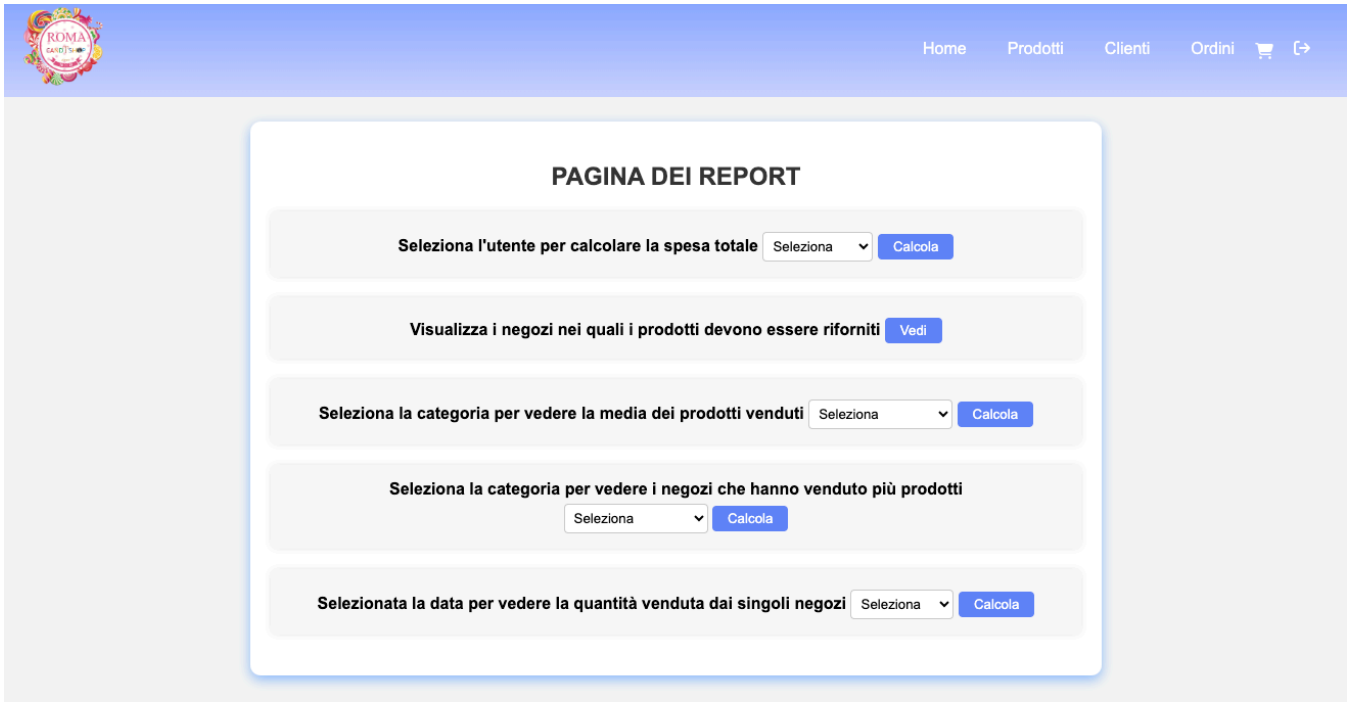


Figura 8: Pagina Report

1° QUERY

La query permette di calcolare la spesa totale effettuata dall'utente selezionato; in particolare, la selezione può essere effettuata soltanto per gli utenti che hanno fatto un acquisto e per i quali si può quindi calcolare la spesa totale. La query è così realizzata:

```
SELECT SUM(S.Importo) AS SpesaTotale
FROM Spesa S JOIN Utente U ON U.Username = S.Username
WHERE U.Username = :username
GROUP BY U.Username
```

Figura 9: Query 1

Mediante la clausola *SELECT* viene applicata la funzione d'aggregazione *SUM* che permette di effettuare la somma del campo importo il quale viene successivamente rinominato con la keyword *AS* in spesaTotale. Nella clausola *FROM* si specificano le tabelle dalle quali devono essere prelevati i dati e in particolare si applica anche una ridenominazione delle tabelle in modo da sfruttare la notazione *nomeTabella.nomeCampo.* Inoltre, al suo interno è presente anche un *JOIN* esplicito tra le due tabelle che viene effettuato sulla condizione posta dopo la keyword *ON* in modo da ottenere soltanto i record che sono semanticamente legati tra loro. In alternativa, si può considerare un *JOIN* implicito che, invece, viene effettuato all'interno della clausola *WHERE*. Sempre all'interno di questa clausola viene applicata la condizione che lo username scelto deve corrispondere con quello presente nei record della tabella Utente. Infine, viene effettuato un raggruppamento sullo username espresso mediante la clausola *GROUP BY* che permette di aggregare i risultati per ogni utente.

Un esempio di come viene visualizzato il risultato dopo aver selezionato l'utente *admin*:

Spesa totale dell'utente : 98.5 €

Figura 10: Esempio di risultato della query 1°

2°QUERY

La query permette di mostrare all'amministratore i negozi i cui prodotti hanno una quantità in magazzino minore rispetto alle scorte minime e che quindi devono essere riforniti.
La query è così realizzata:

```
SELECT N.Nome AS 'Nome Negozio', N.CodiceNegozio AS 'Codice Negozio', P.Nome AS 'Nome Prodotto'
FROM Negozio N JOIN Vendita V ON N.CodiceNegozio = V.CodiceNegozio JOIN Prodotto P ON V.Nome = P.Nome
WHERE P.Nome IN (
    SELECT P.Nome
    FROM Prodotto P JOIN Categoria C ON P.CodiceCategoria = C.CodiceCategoria
    WHERE P.QuantitaMagazzino < C.ScorteMinime
    )
```

Figura 11: Query 2°

Si tratta di una query nidificata cioè di una query all'interno della quale è presente un'altra query che in questo caso è posta all'interno della clausola *WHERE*. La query più interna permette di selezionare i nomi dei prodotti la cui quantità in magazzino è inferiore rispetto alle scorte minime. La query esterna, invece, permette di selezionare il nome del negozio, il codice del negozio e il nome del prodotto per i prodotti che sono presenti all'interno della relazione restituita dalla query più interna. La query nidificata può essere anche riscritta come un'unica query nel seguente modo:

```
SELECT N.Nome AS 'Nome Negozio', N.CodiceNegozio AS 'Codice Negozio', P.Nome AS 'Nome Prodotto'
FROM Negozio N JOIN Vendita V ON N.CodiceNegozio = V.CodiceNegozio JOIN Prodotto P ON V.Nome = P.Nome
    JOIN Categoria C ON P.CodiceCategoria = C.CodiceCategoria
WHERE P.QuantitaMagazzino < C.ScorteMinime
```

Figura 12: Ridefinizione della query 2°

In particolare, i campi selezionati vengono posti nella clausola *SELECT*, le tabelle utilizzate nella query interna sono riportate nella clausola *FROM* con la condizione di *JOIN* e le condizioni vengono poste all'interno della clausola *WHERE*. Di seguito un esempio di come viene visualizzato il risultato:

Nome Negozio	Codice Negozio	Nome Prodotto
CandyShop Roma	007	BonBon Chic
CandyShop Iseo	001	Dolce Spuma
CandyShop Roma	007	Dolce Spuma
CandyShop Torino	008	Dolce Spuma
CandyShop Roma	007	Marshmallow Cake
CandyShop Milano	005	Zucchero Box

Figura 13: Esempio di risultato della query 2°

3° QUERY

La query permette di calcolare la quantità media venduta per la categoria selezionata per ogni negozio interessato; la query è così realizzata:

```
SELECT ROUND(AVG(V.QuantitaVenduta), 2) AS 'Quantità Venduta', N.Nome, V.CodiceNegozio AS 'Codice Negozio'
FROM Vendita V, Negozio N
WHERE V.CodiceNegozio = N.CodiceNegozio AND V.Nome IN (SELECT P.Nome
                                                       FROM Prodotto P
                                                       WHERE P.CodiceCategoria =:categoria)
GROUP BY V.CodiceNegozio
```

Figura 14: Query 3°

Si tratta di una query nidificata dove nella query interna vengono ricavati i nomi dei prodotti che appartengono alla categoria selezionata mentre nella query esterna, per i prodotti venduti presenti all'interno della relazione restituita dalla query interna, viene calcolata la media dei prodotti venduti per ogni negozio. Inoltre, la funzione *ROUND* permette di arrotondare il risultato della funzione d'aggregazione *AVG* a due cifre decimali.

Di seguito un esempio di come viene visualizzato il risultato per la categoria "*torte di caramelle*":

Quantità Venduta	Nome	Codice Negozio
4.00	CandyShop Iseo	001
1.50	CandyShop Bergamo	003
2.00	CandyShop Venezia	004
1.00	CandyShop Milano	005
1.00	CandyShop Napoli	006
1.50	CandyShop Roma	007

Figura 15: Esempio di risultato della query 3°

4° QUERY

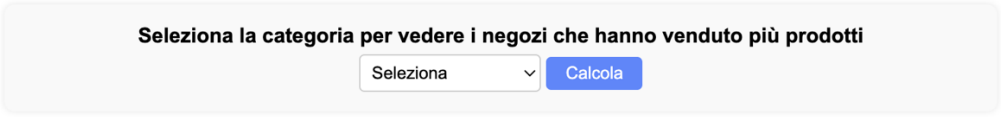
La query permette di restituire il negozio che ha venduto la quantità massima di prodotti per la categoria selezionata; la query viene così realizzata:

```
SELECT DISTINCT N.Citta, N.Nome, V.QuantitaVenduta AS 'Massima Quantità'
FROM Vendita V JOIN NEGOZIO N ON V.CodiceNegozio = N.CodiceNegozio
WHERE V.Nome IN (SELECT P.Nome
                 FROM Prodotto P
                 WHERE P.CodiceCategoria =:categoria)
AND V.QuantitaVenduta = (SELECT MAX(V2.QuantitaVenduta)
                         FROM Vendita V2 JOIN Prodotto P ON V2.Nome = P.Nome
                         WHERE P.CodiceCategoria = :categoria)
```

Figura 16: Query 4°

Si tratta di una query nidificata e in particolare, la prima query interna permette di recuperare il nome dei prodotti della categoria selezionata mentre la seconda query interna restituisce la quantità massima dei prodotti venduti della categoria selezionata.

Invece, la query principale seleziona il nome della città, il nome del negozio e la quantità venduta dei prodotti venduti che sono contenuti nella prima query interna e delle quantità vendute che sono contenute nella seconda query interna. La keyword *DISTINCT* permette di selezionare tuple distinte e quindi di non considerare eventuali duplicati. Di seguito un esempio di come viene visualizzato il risultato:



Citta	Nome	Massima Quantità
Iseo	CandyShop Iseo	2
Bergamo	CandyShop Bergamo	2
Napoli	CandyShop Napoli	2
Roma	CandyShop Roma	2

Figura 17: Esempio di risultato della query 4°

5° QUERY

La query permette di visualizzare i negozi che hanno venduto più prodotti nella data selezionata; la query viene realizzata come di seguito:

```
SELECT N.Nome, SUM(V.QuantitaVenduta) AS 'Quantita Venduta'
FROM Negozio N JOIN Vendita V ON N.CodiceNegozio = V.CodiceNegozio
WHERE N.CodiceNegozio IN (SELECT S.CodiceNegozio
                          FROM Spesa S
                          WHERE S.DataOrdine = :dataOrdine)
GROUP BY V.CodiceNegozio
```

Figura 18: Query 5°

Si tratta di una query nidificata composta da una query interna che permette di selezionare i negozi la cui data dell'ordine è uguale alla data selezionata. La query esterna restituisce il nome del negozio e la quantità venduta per ogni negozio selezionato dalla query interna.
Di seguito un esempio di come viene visualizzato il risultato:

Nome	Quantita Venduta
CandyShop Roma	14

Figura 19: Esempio di risultato della query 5°

REALIZZAZIONE DEL SITO WEB

Il sito web è stato realizzato utilizzando diversi linguaggi:

- HTML: è il linguaggio di markup utilizzato per definire la struttura delle pagina Web;
- CSS: è il linguaggio utilizzato per definire lo stile e la grafica delle pagine HTML;
- PHP: è un linguaggio di scripting lato server che permette di rendere dinamiche le pagine HTML, gestire le richieste degli utenti e interagire con un database.

Inoltre, per la realizzazione della pagina di visualizzazione dei prodotti con le rispettive immagini sono stati integrati i link CSS e JavaScript del framework Bootstrap. Alcuni di questi stili sono stati personalizzati per adattarsi meglio alle esigenze del progetto.

Oltre alle funzionalità di acquisto già descritte, il sito web permette di effettuare la registrazione di un nuovo profilo, l'accesso ad un profilo esistente e il logout. La password fornita dall'utente non viene memorizzata direttamente all'interno del database ma viene gestita tramite la funzione PHP *password_hash()* che genera un hash sicuro utilizzando algoritmi di hashing avanzati per proteggerla da eventuali attacchi informatici. A seconda del ruolo dell'utente che effettua l'accesso, vengono rese disponibili specifiche funzionalità, accessibili attraverso un menù dinamico.
Per effettuare l'accesso come amministratore e quindi avere piene funzionalità si considerano le seguenti credenziali:

- username: admin;
- password: admin.

Nelle pagine di visualizzazione dei prodotti e dei clienti sono stati introdotti dei filtri che permettono di personalizzare la ricerca dei prodotti e dei clienti in base alle esigenze dell'utente. Queste funzionalità sono state implementate principalmente tramite query SQL di base utilizzando le clausole *SELECT, FROM* e *WHERE.*

Il sito web lascia margini di miglioramento futuri tra cui un potenziamento della grafica generale e l'introduzione di una pagina dedicata ai singoli prodotti in cui possono essere visualizzati anche i prodotti correlati. Attualmente, la gestione dell'acquisto dei prodotti è semplificata in quanto non viene considerata la reale possibilità dell'utente di completare l'acquisto: l'importo viene inserito automaticamente ed è pari al totale della spesa. Invece, la data di consegna viene generata casualmente considerando un range da tre giorni fino a una settimana dalla data dell'ordine. Inoltre, non è stata implementata la gestione dei livelli fedeltà degli utenti, né la possibilità di eliminare singoli prodotti dal carrello.