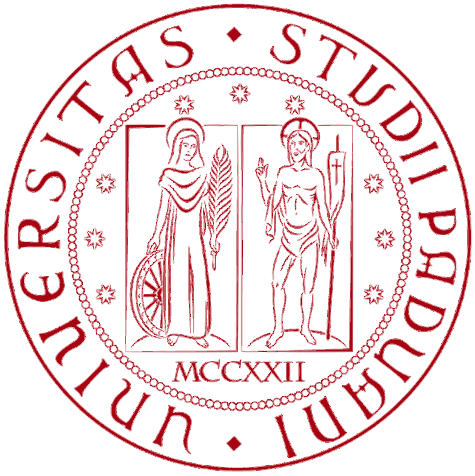
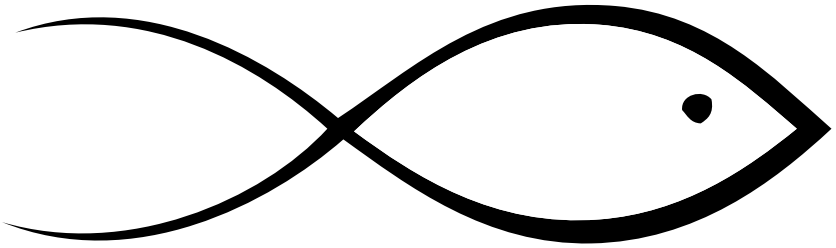


Università degli Studi di Padova

Progetto per il corso di Basi di Dati - A.A. 2012/2013

Pesca eShop



Nicola Carraro mat. 1002050

INDICE

CAPITOLO 1

ABSTRACT

Il progetto consiste nello sviluppare un negozio di pesca online, Pesca eShop, quindi di progettare una base di dati e un sito web per l'interazione degli utenti con essa.

Tutti potranno visualizzare i prodotti presenti, ordinati per tipologia o per marca, ognuno avente una propria scheda dettagliata con foto, descrizione, prezzo e disponibilità.

Ci sono poi gli utenti con un proprio account che si dividono in due categorie:

- Utenti registrati, che hanno un carrello personale al quale possono aggiungere i prodotti che intendono acquistare e che potranno poi comprare acquistando la merce presente.
- Utenti amministratori, che gestiscono la parte amministrativa. Aggiungono nuovi prodotti, controllano la disponibilità residua, gestiscono gli ordini degli utenti confermando gli acquisti da loro eseguiti.

CAPITOLO 2
ANALISI DEI REQUISITI

Il progetto prevede un'interfaccia web per il negozio con tre sezioni principali: una per visionare i prodotti, vedere i loro dettagli e aggiungerli al proprio carrello; un'altra per gestire i prodotti presenti nel carrello e confermare l'acquisto. Una per la sezione amministrativa, cioè una per la gestione della merce, quindi aggiungere i prodotti esauriti o in esaurimento, modificarne i dettagli, come la descrizione o il prezzo, e inserirne di nuovi, e per confermare gli acquisti dei clienti. Ci sarà poi un'altra sezione per permettere il login degli utenti attraverso username e password e una quinta per la registrazione e la modifica dell'account.

Saranno presenti quindi due tipi di utenti: gli utenti registrati, cioè i compratori, che hanno la possibilità di accedere al sito attraverso un username e una password e che possono acquistare i prodotti desiderati dopo averli precedentemente inseriti nel proprio carrello, e gli amministratori, i dipendenti del negozio, i quali, accedendo sempre con un username e una password, avranno la possibilità di modificare i prodotti come scritto sopra.

Per i prodotti interessa sapere il nome, il prezzo, la marca, la tipologia e una breve descrizione. Ci saranno delle classi dedicate per la marca e per la tipologia dei prodotti, perché queste saranno le stesse per merci diverse. Sarà fornita inoltre la quantità residua del prodotto, così da informare l'utente se è possibile o meno acquistarla e così da permettere agli amministratori di verificare le disponibilità del magazzino. Sia per le marche che per le tipologie basterà sapere il nome che servirà a identificarle.
Gli utenti, come già detto, sono di due tipi, registrato e amministratore. Entrambi avranno un nome utente, una password, nome, cognome ed email. L'utente amministratore avrà poi il codice matricola dell'azienda, mentre l'utente registrato avrà il numero della carta di credito e l'indirizzo, con città, CAP e provincia. Questi ultimi tre attributi apparterranno alla classe distinta città.

A ogni utente registrato sarà assegnato un carrello, che costituirà una classe a sé stante, del quale ci servirà sapere il codice identificativo, quali prodotti sono presenti e in che quantità. Sarà presente inoltre una classe acquisti che rappresenterà sia gli acquisti confermati dagli amministratori che quelli ancora in attesa di conferma. Ogni acquisto sarà relativo a un carrello da cui sarà

quindi possibile ricavare i prodotti acquistati. Questa classe conterrà inoltre la data in cui è stato eseguito l'acquisto da parte del cliente, il nome del cliente e il nome dell'amministratore, che potrà anche non esserci nel caso in cui l'acquisto non sia stato confermato dal lato amministrativo.

CAPITOLO 3
PROGETTAZIONE CONCETTUALE

3.1 Classi

Prodotti

Questa classe rappresenta l'insieme dei prodotti in vendita nel negozio, anche se la loro quantità è zero e quindi non sono disponibili. Tutti hanno una marca e una tipologia che verranno memorizzate in classi esterne essendo comuni a più prodotti.

Attributi

- *CodiceID: int «PK»* - codice identificativo del prodotto. Chiave primaria
- *Nome: string* - nome del prodotto
- *Prezzo: real* - prezzo dell'oggetto
- *Quantita: int* - quantità residua del prodotto in magazzino
- *Descrizione: string* - descrizione del prodotto
- *Immagine: string* - URL dell'immagine

Marche

Questa classe rappresenta l'insieme delle marche dei prodotti.

Attributi

- *Nome: string «PK»* - nome della marca. Chiave primaria

Tipologie

Questa classe rappresenta l'insieme delle tipologie dei prodotti.

Attributi

- *Nome: string «PK»* - nome della tipologia. Chiave primaria

Carrelli

Questa classe rappresenta l'insieme dei carrelli degli utenti registrati. Ogni utente possiede un carrello nel quale può aggiungere i prodotti che intende acquistare.

Attributi

- *CodiceID: int «PK»* - codice identificativo del carrello. Chiave primaria

Acquisti

Questa classe rappresenta l'insieme degli acquisti effettuati dai clienti sia in attesa di conferma da parte degli amministratori sia già confermati. Conterrà le informazioni su chi è l'utente che ha fatto l'acquisto e quando lo ha fatto e chi è l'amministratore che lo ha confermato.

Attributi

- *CodiceID: int «PK»* - codice identificativo dell'acquisto. Chiave primaria
- *Data: date* - data in cui il cliente ha eseguito l'acquisto

Utenti

Questa classe rappresenta l'insieme degli utenti che possono effettuare delle operazioni sul sito, quindi sulla base di dati. Essendo una gerarchia, ci sono la classe Utenti, che ha gli attributi comuni, e le altre due classi che possiedono attributi specifici per ognuna.

Attributi

- *NomeUtente: string «PK»* - Nome utente identificativo per ciascuno e con il quale accederanno alla propria area riservata. Chiave primaria
- *Password: string* - Password per l'accesso all'area personale
- *Nome: string* - Nome dell'utente
- *Cognome: string* - Cognome dell'utente
- *Email: string* . Email dell'utente
- *Registrati* - Classe che rappresenta gli utenti che possono eseguire gli acquisti. Conterrà anche il riferimento al carrello in uso
  - *Indirizzo: string* - Indirizzo dell'utente
  - *Carta: string* - Numero carta di credito dell'utente
- *Amministratori* - Classe che rappresenta gli utenti amministratori che possono gestire le merci e confermare gli acquisti dei clienti
  - *Matricola: int* - Matricola aziendale identificativa

Citta

Questa classe rappresenta l'insieme delle città degli utenti registrati.

Attributi

- *CodiceID: int «PK»* - Codice identificativo della città. Chiave primaria
- *Nome: string* - Nome della citta
- *Provincia: string* - Provincia

3.2 Associazioni

Marche - Prodotti: marcheprodotti

Associazione tra Marche e Prodotti. Ogni marca può avere uno più prodotti, ogni prodotto è di una marca. Ha molteplicità 1-M. É parziale da Marche a Prodotti ed è totale anche tra Prodotti e Marche.

Tipologie - Prodotti: tipologieprodotti

Associazione tra Tipologie e Prodotti. Ogni tipologia può avere uno o più prodotti, ogni prodotto è di una certa tipologia. Ha molteplicità 1-M. É parziale da Tipologie a Prodotti, è totale tra Prodotti e Tipologie.

Carrelli - Prodotti: contiene

Associazione tra Carrelli e Prodotti. Un carrello può contenere uno o più prodotti, un prodotto può essere contenuto in uno o più carrelli. Ha molteplicità M-N. É parziale da Carrelli a Prodotti, come lo è tra Prodotti e Carrelli.
Questa associazione ha la proprietà di quantità per indicare quanti prodotti dello stesso tipo sono presenti all'interno del carrello.

Registrati - Carrelli: possiede

Associazione tra Registrati e Carrelli. Un registrato possiede un carrello, un carrello è posseduto da un solo registrato. Ha molteplicità 1-1. É totale in entrambi i sensi.

Acquisti - Carrelli: riguarda

Associazione tra Acquisti e Carrelli. Un acquisto riguarda un carrello, un carrello può essere oggetto di un acquisto. Ha molteplicità 1-1. É parziale in entrambi i sensi.

Acquisti - Registrati: eseguito da

Associazione tra Acquisti e Registrati. Un acquisto è eseguito da un registrato. Un registrato può eseguire uno o più acquisti. Ha molteplicità M-1. É totale in entrambi i sensi.

Acquisti - Amministratori: confermato da

Associazione tra Acquisti e Amministratori. Un acquisto è confermato da un amministratore. Un amministratore conferma uno o più acquisti. Ha molteplicità M-1. É parziale da Acquisti ad Amministratori, totale per il senso opposto.

Registrati - Citta: abita

Associazione tra Registrati e Città. Un utente registrato abita in una città, in una città ci abita nessuno o almeno un utente. Ha molteplicità 1-M. É totale tra registrato e città, e parziale tra città e registrato.

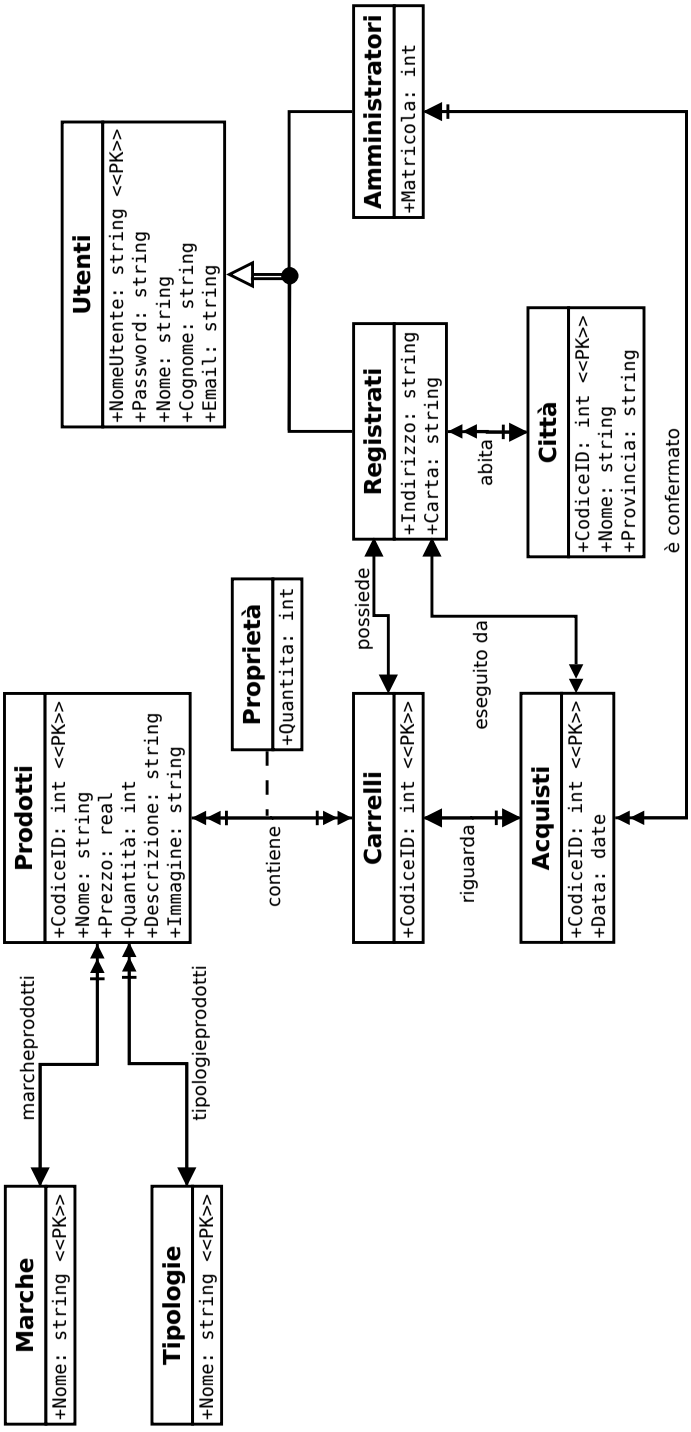
3.3 Gerarchia delle classi

Utenti: Registrati - Amministratori

La gerarchia di utenti permette di avere gli attributi comuni dei registrati e degli amministratori nella classe Utenti e gli attributi specifici nelle rispettive classi.

- Classi disgiunte
- L'unione è una copertura

3.4 Schema Concettuale



CAPITOLO 4
PROGETTAZIONE LOGICA

4.1 Classi

Prodotti

- CodiceID «PK»
- Nome
- Prezzo
- Quantita
- *Marca «FK»: string* - chiave esterna a Nome della classe Marche
- *Tipologia «FK»: string* - chiave esterna a Nome della classe Tipologie
- Descrizione
- Immagine

Marche

- Nome «PK»

Tipologie

- Nome «PK»

Contiene

Classe aggiunta per l'associazione M-N tra Prodotti e Carrelli. Include la proprietà di quantità.

- *Prodotto «PK» «FK»: int* - chiave esterna a CodiceID della classe Prodotti

- *Carrello «PK» «FK»: int* - chiave esterna a CodiceID della classe Carrelli
- *Quantita: int «NOT NULL»* - quantità del prodotto presente nel carrello

Carrelli

- CodiceID «PK»

Acquisti

- CodiceID «PK»
- Data
- *Carrello «FK»: int «NOT NULL»* - chiave esterna a CodiceID della classe Carrelli
- *Registrato «FK»: string* - chiave esterna a NomeUtente della classe Registrati
- *Amministratore «FK»: string* chiave esterna a NomeUtente della classe Amministratori

Utenti

- NomeUtente «PK»
- Password «NOT NULL»
- Nome
- Cognome
- Email

Registrati

- *NomeUtente «PK» «FK»: string* - chiave esterna a NomeUtente della classe Utenti. Chiave Primaria
- Indirizzo
- *Citta «FK»: int* - chiave esterna a CodiceID della classe Citta
- Carta
- *Carrello «FK»: int «NOT NULL* - chiave esterna a CodiceID della classe Carrelli

Amministratori

- *NomeUtente «PK» «FK»: string* - chiave esterna a NomeUtente della classe Utenti. Chiave Primaria
- Matricola

Citta

- CodiceID «PK»
- Nome
- Provincia

4.2 Considerazioni progettuali

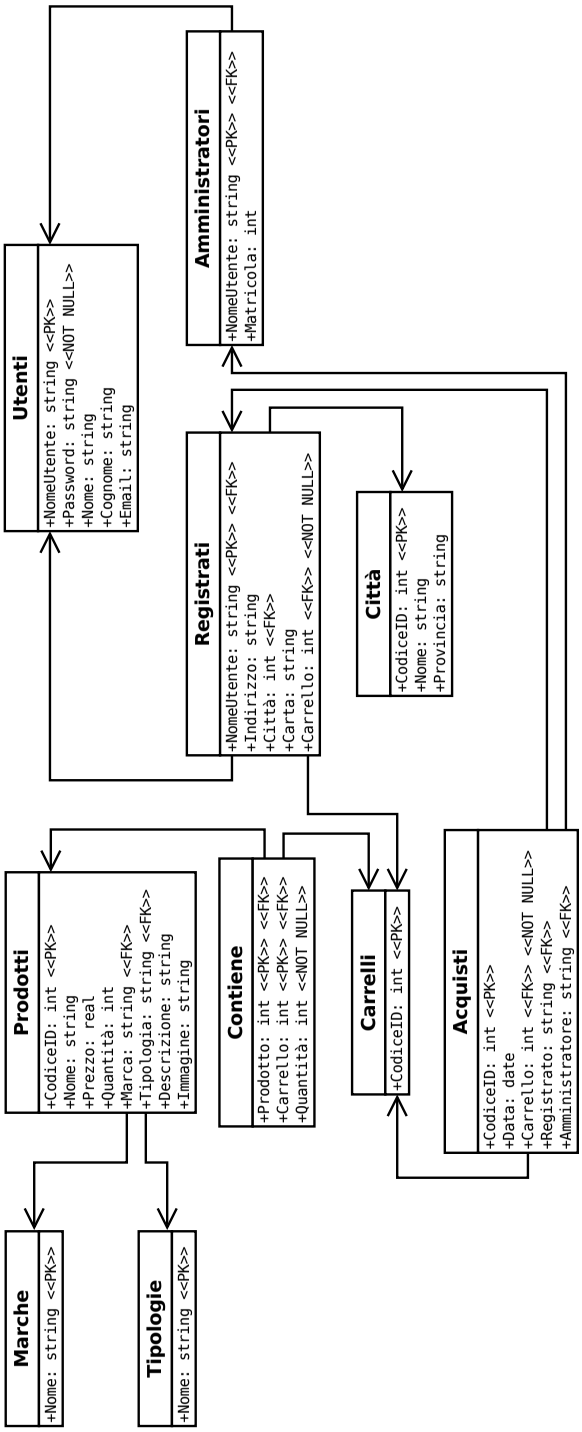
Si assume che ci sia un numero di carrelli pari al numero di utenti registrati più il numero di acquisti effettuati. Ciò perché nel momento in cui un utente conferma l'acquisto di un carrello, questo viene registrato tra gli acquisti e all'utente viene assegnato un altro carrello con codice identificativo differente. In questo modo si riesce a tenere traccia degli acquisti effettuati dagli utenti in passato e risalire inoltre all'amministratore che l'ha confermato.

Nella classe Contiene l'attributo Quantita non può essere NULL in quanto altrimenti perderebbe il significato. Un prodotto infatti è presente nel carrello se ha almeno quantità 1.

Un record della classe Acquisti non può avere il campo Carrello NULL altrimenti non avrebbe senso di esistere. Gli acquisti infatti sono le conferme dei carrelli una volta proceduto con il pagamento di essi. Possono avere il campo registrato o amministratore vuoto, invece, in quanto nel primo caso servirà per tenere traccia degli acquisti gestiti dagli amministratori anche se l'utente non è più registrato al sito, mentre il secondo caso si verifica perché gli acquisti vengono presi in carico solo dopo la conferma del cliente.

L'attributo Carta nella classe Registrati è di tipo string nonostante sia composto solo di numeri per ottimizzare l'utilizzo della memoria in quanto non sono previste operazioni da fare con esso.

4.3 Schema Logico



CAPITOLO 5

DEFINIZIONE SCHEMA LOGICO CON IL DDL MYSQL

Listing 5.1: Tabella Marche

```
CREATE TABLE Marche (
        Nome VARCHAR (20) PRIMARY KEY
) ENGINE= InnoDB;
```

Listing 5.2: Tabella Tipologie

```
CREATE TABLE Tipologie (
        Nome VARCHAR (20) PRIMARY KEY
) ENGINE= InnoDB;
```

Listing 5.3: Tabella Prodotti

```
CREATE TABLE Prodotti (
        CodiceID INT PRIMARY KEY,
        Nome VARCHAR (20),
        Prezzo REAL,
        Quantita INT DEFAULT 0,
        Marca VARCHAR (20),
        Tipologia VARCHAR (20),
        Descrizione VARCHAR (100),
        Immagine VARCHAR (20),

        FOREIGN KEY (Marca) REFERENCES Marche (Nome)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,

        FOREIGN KEY (Tipologia) REFERENCES Tipologie (Nome)
                ON DELETE NO ACTION
                ON UPDATE CASCADE
) ENGINE= InnoDB;
```

Listing 5.4: Tabella Carrelli

```
CREATE TABLE Carrelli (
        CodiceID INT PRIMARY KEY
) ENGINE= InnoDB;
```

Listing 5.5: Tabella Contiene

```
CREATE TABLE Contiene (
        Prodotto INT,
        Carrello INT,
        Quantita INT NOT NULL,

        PRIMARY KEY (Prodotto, Carrello),

        FOREIGN KEY (Prodotto) REFERENCES Prodotti (CodiceID
            )
                ON DELETE CASCADE
                ON UPDATE CASCADE,

        FOREIGN KEY (Carrello) REFERENCES Carrelli (CodiceID
            )
                ON DELETE CASCADE
                ON UPDATE CASCADE
) ENGINE= InnoDB;
```

Listing 5.6: Tabella Utenti

```
CREATE TABLE Utenti (
        NomeUtente VARCHAR (20) PRIMARY KEY,
        Password VARCHAR (20) NOT NULL,
        Nome VARCHAR (30),
        Cognome VARCHAR (30),
        Email VARCHAR (50)
) ENGINE= InnoDB;
```

Listing 5.7: Tabella Amministratori

```
CREATE TABLE Amministratori (
        NomeUtente VARCHAR (20) PRIMARY KEY,
        Matricola INT,

        FOREIGN KEY (NomeUtente) REFERENCES Utenti (
            NomeUtente)
                ON DELETE CASCADE
                ON UPDATE CASCADE
) ENGINE= InnoDB;
```

Listing 5.8: Tabella Città

```
CREATE TABLE Citta (
        CodiceID INT PRIMARY KEY,
        Nome VARCHAR (20),
        Provincia CHAR (2)
) ENGINE= InnoDB;
```

Listing 5.9: Tabella Registrati

```
CREATE TABLE Registrati (
        NomeUtente VARCHAR (20) PRIMARY KEY,
        Indirizzo VARCHAR (50),
        Citta INT,
        Carta CHAR(16),
        Carrello INT NOT NULL,

        FOREIGN KEY (NomeUtente) REFERENCES Utenti (
            NomeUtente)
                ON DELETE CASCADE
                ON UPDATE CASCADE,

        FOREIGN KEY (Citta) REFERENCES Citta (CodiceID)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,

        FOREIGN KEY (Carrello) REFERENCES Carrelli (CodiceID
            )
                ON DELETE NO ACTION
                ON UPDATE CASCADE
) ENGINE= InnoDB;
```

Listing 5.10: Tabella Acquisti

```
CREATE TABLE Acquisti (
        CodiceID INT PRIMARY KEY,
        Data DATE,
        Carrello INT NOT NULL,
        Registrato VARCHAR (20),
        Amministratore VARCHAR (20),

        FOREIGN KEY (Carrello) REFERENCES Carrelli (CodiceID
            )
                ON DELETE NO ACTION
                ON UPDATE CASCADE,

        FOREIGN KEY (Registrato) REFERENCES Registrati (
            NomeUtente)
                ON DELETE SET NULL
                ON UPDATE CASCADE,

        FOREIGN KEY (Amministratore) REFERENCES
            Amministratori (NomeUtente)
                ON DELETE SET NULL
                ON UPDATE CASCADE
) ENGINE= InnoDB;
```

Listing 5.11: Tabella Errori

```
CREATE TABLE Errori (
        CodiceID INT PRIMARY KEY
) Engine=InnoDB;
```

CAPITOLO 6
QUERY, FUNZIONI E TRIGGER

6.1 Query

Query 1

Listing 6.1: query1

```
1 SELECT count(*) AS NCarrelli
2 FROM Carrelli c JOIN Acquisti a ON c.CodiceID= a.Carrello
3 WHERE a.Amministratore IS NOT NULL AND a.Data>= "2012-06-01"
      AND a.Data<= "2012-12-31";
```

La query dà in output il numero di acquisti confermati dagli amministratori in un semestre, in questo caso nell'ultimo semestre dell'anno 2012. L'output generato con la corrente base di dati è:

Listing 6.2: query1 Risultati

```
+-----------+
| NCarrelli |
+-----------+
|         1 |
+-----------+
```

Query 2

Listing 6.3: query2

```
1 CREATE VIEW sommaTotali (CodiceID, Totale) AS
2         SELECT ca.CodiceID, SUM(p.Prezzo*co.Quantita)
3         FROM Prodotti p JOIN Contiene co ON(p.CodiceID=co.
              Prodotto)
4                 JOIN Carrelli ca ON (co.Carrello=ca.CodiceID
                      )
5         WHERE ca.CodiceID IN (
6                 SELECT Carrello
```

```
7                   FROM Acquisti)
8           GROUP BY ca.CodiceID;
9
10  SELECT r.NomeUtente, max(st.Totale)
11  FROM sommaTotali st JOIN Acquisti a ON (st.CodiceID=a.
        Carrello) JOIN Registrati r ON (a.Registrato=r.NomeUtente
        )
12  GROUP BY r.NomeUtente;
```

La query fornisce in output la spesa di valore più alto effettuata da ciascun utente dal momento in cui si è registrato. L'output generato è:

Listing 6.4: query2 Risultati

```
+-------------+---------------+
| NomeUtente  | SpesaMaggiore |
+-------------+---------------+
| chiaraCarpa |          1650 |
| federico91  |         377.4 |
| tonib       |           675 |
+-------------+---------------+
```

Query 3

Listing 6.5: query3

```
1   CREATE VIEW contaProdotti (Nome, Conta) AS
2           SELECT p.Nome, SUM(co.Quantita)
3           FROM Acquisti a JOIN Carrelli ca ON (a.Carrello=ca.
                CodiceID)
4                   JOIN Contiene co ON (ca.CodiceID=co.Carrello
                        )
5                   JOIN Prodotti p ON (co.Prodotto=p.CodiceID)
6           GROUP BY p.CodiceID;
7
8   SELECT Nome, Conta
9   FROM contaProdotti;
10  WHERE Conta= (SELECT max(Conta) FROM contaProdotti);
```

La query sopra riportata ritorna il prodotto più venduto del negozio. L'output generato in questo caso riporta due prodotti, in quanto di entrambi è stata venduta la stessa quantità che corrisponde a quella massima.

Listing 6.6: query3 Risultati

```
+----------------+-------+
| Nome           | Conta |
+----------------+-------+
| Stiletto       |     5 |
| Boiles Fragola |     5 |
+----------------+-------+
```

Query 4

Listing 6.7: query4

```
1  SELECT CodiceID, Nome
2  FROM Prodotti
3  WHERE CodiceID NOT IN (
4          SELECT p.CodiceID
5          FROM Prodotti p JOIN Contiene co ON (p.CodiceID=co.
               Prodotto)
6                  JOIN Carrelli ca ON (co.Carrello=ca.
                       CodiceID)
7                  JOIN Acquisti a ON (ca.CodiceID=a.Carrello)
8          WHERE EXTRACT(YEAR FROM a.Data)>=EXTRACT(YEAR FROM
               DATE_SUB(CURDATE(),INTERVAL 6 MONTH)) AND
9                   EXTRACT(MONTH FROM a.Data)>=EXTRACT(MONTH
                        FROM DATE_SUB(CURDATE(),INTERVAL 6
                        MONTH))
10         );
```

La query restituisce in output la lista dei prodotti (Nome e CodiceID) i quali non sono stati venduti negli ultimi sei mesi. In questa query è stata usata anche la funzione EXTRACT() per estrapolare anno e mese dalla data corrente e dalla data di acquisto, così da poter dare dei risultati sempre aggiornati al momento in cui viene lanciata la query. L'output generato il giorno della consegna è:

Listing 6.8: query4 Risultati

```
+----------+----------------+
| CodiceID | Nome           |
+----------+----------------+
|        2 | Tribal         |
|        3 | Ultegra        |
|        4 | Endless        |
|        5 | NTX-D CAM Mono |
|        6 | Black Metal    |
|        8 | Stiletto       |
|        9 | Boiles Fragola |
|       12 | NTX-R          |
|       13 | Pera           |
|       14 | Arma5          |
|       15 | Piattello      |
+----------+----------------+
```

6.2 Funzioni

Funzione 1

Listing 6.9: Funzione maxCodAcquisto

```
1  CREATE FUNCTION maxCodAcquisto () RETURNS INT
2  BEGIN
3          DECLARE maxCod INT;
```

```
4         SELECT MAX(CodiceID) INTO maxCod FROM Acquisti;
5         RETURN maxCod;
6 END
```

Funzione che ritorna il più grande CodiceID per la tabella Acquisti.

Funzione 2

Listing 6.10: Funzione maxCodCarrello

```
1 CREATE FUNCTION maxCodCarrello () RETURNS INT
2 BEGIN
3         DECLARE maxCod INT;
4         SELECT MAX(CodiceID) INTO maxCod FROM Carrelli;
5         RETURN maxCod;
6 END
```

Funzione che ritorna il più grande CodiceID per la tabella Carrelli.

Funzione 3

Listing 6.11: Funzione maxCodProdotto

```
1 CREATE FUNCTION maxCodProdotto () RETURNS INT
2 BEGIN
3         DECLARE maxCod INT;
4         SELECT MAX(CodiceID) INTO maxCod FROM Prodotti;
5         RETURN maxCod;
6 END
```

Funzione che ritorna il più grande CodiceID per la tabella Prodotti.

Funzione 4

Listing 6.12: Funzione totaleCarrello

```
1  CREATE FUNCTION totaleCarrello (codID INT) RETURNS DECIMAL
       (4,2)
2  BEGIN
3         DECLARE tot DECIMAL (4,2);
4
5         SELECT SUM(p.Prezzo*co.Quantita) INTO tot
6         FROM Prodotti p JOIN Contiene co ON p.CodiceID=co.
              Prodotto JOIN Carrelli ca ON co.Carrello=ca.
              CodiceID
7         WHERE ca.CodiceID=codID;
8
9         RETURN tot;
10 END
```

Funzione che ritorna il totale del valore dei prodotti del carrello, moltiplicando il costo di ciascun prodotto per la rispettiva quantità presente nel carrello.

Funzione 5

Listing 6.13: Funzione verificaTipoUtente

```
1  CREATE FUNCTION verificaTipoUtente (nomeUt VARCHAR(10))
       RETURNS CHAR(1)
2  BEGIN
3          DECLARE test VARCHAR(10);
4          DECLARE tipo CHAR(1);
5
6          SELECT NomeUtente INTO test
7          FROM Utenti NATURAL JOIN Registrati
8          WHERE NomeUtente= nomeUt;
9
10         IF (test IS NOT NULL) THEN SET tipo= 'r';
11
12         ELSE
13                 SET tipo= 'a';
14
15         END IF;
16
17         RETURN tipo;
18 END
```

Funzione che verifica se l'utente passato come parametro è di tipo Registrato oppure di tipo Amministratore, ritornando r oppure a a seconda del tipo.

6.3 Trigger

Si fa notare che i trigger creati per testare certe situazioni, in caso di un risultato falso, per bloccare il processo in corso inseriscono un record con valore NULL nella tabella Errori. Essendo questo campo chiave, fa si che il database lanci un'eccezione e blocchi conseguentemente l'operazione.

Trigger 1

Listing 6.14: Trigger controlloInserimentoCarrello

```
1  CREATE TRIGGER controlloInserimentoCarrello BEFORE INSERT ON
       Contiene
2  FOR EACH ROW
3  BEGIN
4          DECLARE pro INT;
5
6          SELECT Prodotto INTO pro
7          FROM Contiene c JOIN Prodotti p ON c.Prodotto=p.
               CodiceID
```

```
8           WHERE p.CodiceID= NEW.Prodotto AND c.Carrello= NEW.
                Carrello;
9
10          IF (pro IS NOT NULL) THEN
11                  INSERT INTO Errori VALUE(NULL);
12          END IF;
13
14          IF (NEW.Quantita<0) THEN
15                  INSERT INTO Errori VALUE (NULL);
16          END IF;
17
18  END $
```

Trigger che si attiva prima dell'inserimento di un record nella tabella Contiene, cioè prima di inserire un prodotto nel carrello, per verificare che esso non sia già presente.

Trigger 2

Listing 6.15: Trigger controlloInserimentoProdotto

```
1   CREATE TRIGGER controlloInsProdotto BEFORE INSERT ON
        Prodotti
2   FOR EACH ROW
3   BEGIN
4           DECLARE tip VARCHAR (10);
5           DECLARE mar VARCHAR (10);
6
7           SELECT Nome INTO tip FROM Tipologie WHERE Nome= NEW.
                Tipologia;
8           IF (tip IS NULL) THEN
9                   INSERT INTO Tipologie VALUE(NEW.Tipologia);
10          END IF;
11
12
13          SELECT Nome INTO mar FROM Marche WHERE Nome= NEW.
                Marca;
14          IF (mar IS NULL) THEN
15                  INSERT INTO Marche VALUE(NEW.Marca);
16          END IF;
17
18  END $
```

Trigger che si attiva prima dell'inserimento di un nuovo prodotto, cioè di un nuovo record nella tabella Prodotti. Verifica se la marca e la tipologia a cui esso appartiene sono già presenti nelle rispettive tabelle Marche e Tipologie. Se non è così, aggiunge un record alle tabelle con i valori indicati dal prodotto.

Trigger 3

Listing 6.16: Trigger controlloNomeUtente

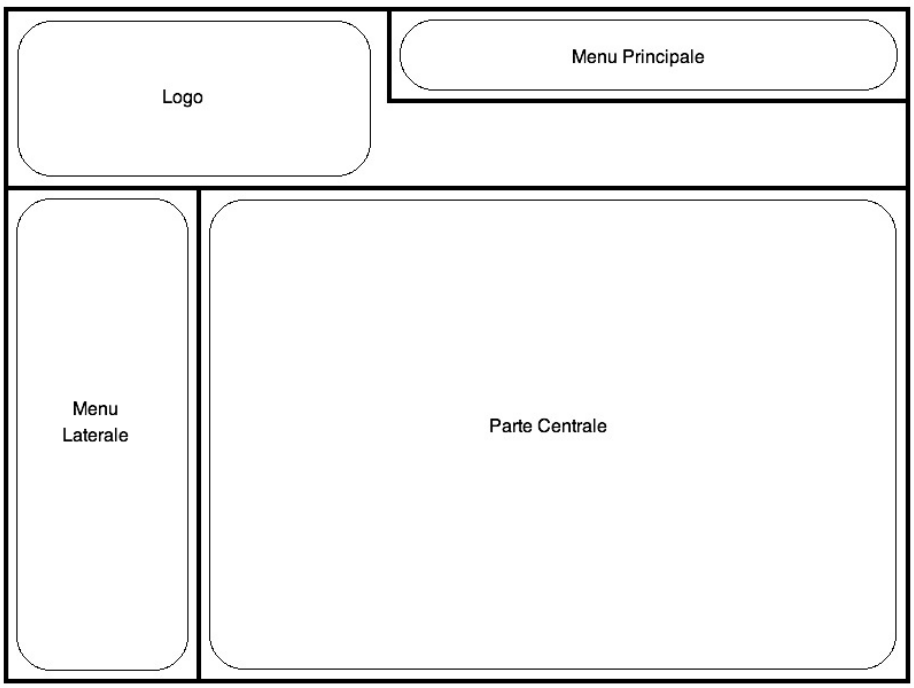
```
1  CREATE TRIGGER controlloNomeUtente BEFORE INSERT ON Utenti
2  FOR EACH ROW
3  BEGIN
4          DECLARE nom VARCHAR (10);
5
6          SELECT NomeUtente INTO nom FROM Utenti WHERE
               NomeUtente= NEW.NomeUtente;
7          IF (nom IS NOT NULL) THEN
8                  INSERT INTO Errori VALUE(NULL);
9          END IF;
10
11 END $
```

Trigger che controlla la presenza o meno del nome utente inserito durante la registrazione. Essendo il campo chiave per la tabella Utenti, se è già presente ritorna un errore.

CAPITOLO 7

INTERFACCIA WEB

L'interfaccia web creata per gestire la base di dati è strutturalmente semplice. Si può schematizzare infatti come nell'immagine che segue:



Dal menu principale si può accedere in qualunque momento alle pagine principali del sito. Nella parte centrale si svolgeranno le varie operazioni e saranno visualizzati i vari record. Il menu laterale, invece, conterrà opzioni diverse a seconda della pagina e del tipo di utente che ha eseguito il login.

Il sito è composto, quindi, da una *home page* in cui ci sono le istruzioni per il sito. Una pagina *prodotti* nella quale è possibile visualizzare i prodotti contenuti in magazzino filtrandoli per la categoria scelta dal menu laterale e aggiungerli al

proprio carrello dopo aver inserito la quantità desiderata. Una pagina *Account* in cui ogni utente può visualizzare tutti i suoi dati e modificarli. Una pagina *Carrello* nella quale gli utenti Registrati possono visualizzare i prodotti contenuti nel proprio carrello, eventualmente rimuoverli, verificare il prezzo totale e confermare lo stesso. Una pagina *amministrazione*, che permette l'accesso a varie funzioni per gli utenti amministratori, come l'aggiunta e l'eliminazione di un prodotto, l'aggiunta di altri amministratori, la visualizzazione e la conferma degli ordini.

7.1 Descrizione pagine

Le pagine utilizzano alcuni tra le query, le funzioni e i trigger descritti sopra. In generale, comunque, la maggior parte delle pagine sono pagine php, fatta eccezione per quelle del login, della registrazione e dell'inserimento. Questo perché le pagine necessitano di connettersi al database per eseguire delle query su di esso. Alcune pagine sono poi state create solamente per eseguire certe operazioni, così da dividere la parte logica dalla struttura del sito.

In particolare la pagina *lib.php* contiene tutte le funzioni necessarie alle pagine eseguire le operazioni più ripetitive.

Qui di seguito sono descritte le pagine più nel dettaglio, indicando le loro relazioni con le pagine di tipo "logico".

7.1.1 Pagine principali

La pagina *index.php* è la pagina iniziale del sito. Le pagine *prodotti.php*, *account.php*, *carrello.php* e *amministrazione.php* sono le altre pagine del sito, citate anche in precedenza.

7.1.2 Pagine gestione accesso

La pagina *login.html* è la pagina che fornisce un form in cui è possibile inserire il nome utente e la password per accedere al sito. É presente un link in ogni pagina per poter accederci in qualunque momento. Le pagine *registrazione.html* e *registrazioneadmin.html* forniscono invece un form contenente tutti i campi da compilare per poter registrarsi al sito come utente Registrato e Amministratore rispettivamente. La pagina *logout.php* viene aperta nel momento in cui un utente decide di disconnettersi. Distrugge la sessione attiva e reindirizza l'utente alla home.

Le pagine appena citate si avvalgono di altre pagine php che svolgono le funzioni necessarie ad attuare le operazioni richieste dall'utente. In particolare,la pagina *loginmanage.php* controlla che il nomeutente e la password inseriti corrispondano ai dati presenti nel database e, se così, inizia una nuova sessione.

La pagina *addaccount.php* interagisce con il database per aggiungere un utente Registrato con tutti i dati inseriti dall'utente, creando inoltre anche un nuovo carrello e associandolo a esso, mentre la pagina *addaccountadmin.php* inserisce nel database un utente amministratore. La pagina *updateaccount.php* svolge le operazioni necessarie per aggiornare i dati dell'utente, differenziando a seconda se questo è un utente Registrato o un utente Amministratore.

7.1.3 Pagine gestione prodotti

Per la gestione dei prodotti sono state create alcune pagine che permettono l'inserimento delle informazioni e altre usate allo scopo di eseguire le operazioni in background che agiscono nel database. La pagina *nuovoprodotto.html* fornisce quindi il form per l'inserimento dei dati di un nuovo prodotto, mentre *aggiornaprodotto.php* il form per aggiornare i dettagli. Queste pagine chiamano rispettivamente le pagine *addproduct.php* e *updateproduct.php* le quali agiscono sul database inserendo un record o aggiornandolo.

C'è una terza pagina di livello logico utile alla gestione dei prodotti che è *deleteproduct.php*, la quale rimuove il prodotto dal database dopo che è stato fornito il suo CodiceID.

7.1.4 Pagine gestione carrello

Ci sono altre tre pagine di livello logico che permettono la gestione del carrello per inserire e rimuovere i prodotti in esso (*addtokart.php* e *removefromkart.php* e per confermarlo *confirmkart.php*, così da aggiungerlo tra gli Acquisti e assegnare un nuovo carrello all'utente.

7.1.5 Pagine gestione acquisti

É presente infine una pagina per la conferma degli acquisti da parte degli amministratori. *confirmpurchase.php* permette infatti di aggiornare i record contenuti nella tabella acquisti così da farli risultare confermati anche dagli amministratori.

CAPITOLO 8

NOTE

Il materiale consegnato per il progetto comprende:

- Relazione
- Cartella *eShop*: pagine dell'interfaccia web, comprese anche le pagine sorgenti in php
- Cartella *Sql*: file sorgenti sql per creare il database, file di testo per popolare le tabelle, file delle query, funzioni, trigger.

Il sito disponibile all'indirizzo: ../basidati/∼ ncarraro è impostato per funzionare con il MySql collegato e il database ncarraro-ES. Per modificare le impostazioni di connessione al server è necessario modificare il file *lib.php* contenuto nella cartella eShop/php, in particolare la funzione connessioneDB() e le variabili contenute in essa.