MyHair

Università di Padova

Progetto MySql – PHP [BasiDati]

Sebastiano Marchesini

Jennifer Celadon

09/07/2012

Posta elettronica:
sebastiano.marchesini@studenti.math.unipd.it
jennifer.celadon@studenti.math.unipd.it
Web:
http://localhost:20080/basidati/~smarches

Abstract

Il progetto MyHair è strutturato su una base di dati mysql per la gestione di un negozio di parrucchieri, fulcro del programma risiede nei clienti, salvati nel database e collegati agli appuntamenti da loro prenotati.

Il contesto su cui si appoggia la scelta di questo programma riguarda un salone di piccole dimensioni, si parla di un prodotto accessibile da tutte le dipendenti e titolari per tenere aggiornato il negozio con il massimo dell'efficienza.

Una delle sue funzioni principali risiede nella gestione degli appuntamenti, tra i quali troviamo il calcolo della media dei TipoAppuntamento prenotato, la loro relativa modifica e cancellazione, una veloce ricerca della corrispondenza data-cliente e i prodotti più usati all'interno del negozio.

Possibile altra funzione utilizzabile riguarda il contesto Clienti, dove troviamo il controllo dei compleanni in ogni mese, per poter offrire uno sconto applicabile al saldo finale dell'appuntamento e infine la visualizzazione del cliente con maggior numero di frequenza in negozio.

I prodotti vengono gestiti dal programma per la loro eventuale modifica e cancellazione, le risultanti scorte di magazzino e uno storico dei prodotti utilizzati per il colore della tinta applicata ai clienti.

Analisi dei Requisiti

Cliente :

Ogni cliente può avere uno o più appuntamenti oppure nessuno.
Il compleanno di ogni cliente è aggiornato alla data del prossimo compleanno. Ogni cliente ha uno sconto il giorno del suo compleanno, e viene avvisato un mese prima grazie ad un email.

Appuntamento :

Di ogni appuntamento e' importante sapere la data, il codice dell'appuntamento.
Inoltre di un appuntamento si vuole sapere il costo, e deve essere positivo se è un Appuntamento di un Cliente,
e negativo se è un Appuntamento Speciale, cioè di aggiornamento.
Gli Appuntamenti quindi si dividono in due classi, gli appuntamenti con clienti e quelli di aggiornamento.
Bisogna poter ricavare quant'è il guadagno del mese (somma algebrica tra il costo degli appuntamenti clienti, e le spese degli appuntamenti speciali)

AppuntamentiClienti :

Di un appuntamento clienti è importante sapere di chi è, e di che cosa si tratta (il tipo di appuntamento si è prenotato il cliente). Bisogna sapere se il Cliente ha uno sconto da poter usufruire su quell'appuntamento.
Durante un appuntamento si possono usare uno o più prodotti (anche nessuno) in diverse quantità.
Non ci possono essere più appuntamenti lo stesso giorno alla stessa ora dello stesso cliente.

Prodotti :

Ogni prodotto può avere una sola marca, e un solo tipo prodotto.
Un prodotto può essere usato a più appuntamenti in quantità diverse. Bisogna poter sapere quali sono i prodotti in esaurimento tra quelli che vengono ordinati (cioè non quelli che ci sono in magazzino) per poter fare il nuovo ordine.

Classi

Cliente:
CodCliente: int
Nome: VarChar(10)
Cognome: VarChar(10)
Compleanno: Date
Telefono: VarChar(10)
Email: VarChar(50)

Il compleanno contiene la data del prossimo compleanno, quindi il giorno e mese corrispondono a quelli della data di nascita mentre l'anno viene aggiornato ad ogni compleanno, in modo tale che consideri sempre il prossimo compleanno.

Appuntamento:
CodAppuntamento: int
DataOra: DateTime
Costo: Double

Gli appuntamenti Speciali hanno costo negativo, in quanto sono spese a carico del titolare ha per i corsi di aggiornamento, mentre gli appuntamenti Clienti hanno costo positivo da considerarsi in entrata nel resoconto.

AppuntamentiSpeciali:
CodAppuntamento: int
Durata: Time
NomeAppuntamento: VarChar(15)
Luogo: VarChar(20)
LimitePersone: int

AppuntamentiClienti:
CodAppuntamento: int
CodCliente: int
Sconto: Double
TipoAppuntamento:ENUM

Prodotti:
CodProdotto: int
Marca: ENUM('Wella', 'SP', 'Sebastian')
Tipo: ENUM('Koleston Perfect', 'Inspire by KP', 'Welloxon Perfect', 'Color Touch', 'Color Fresh', 'Perfection', 'Eos', 'Blondor Multi-Blonde', 'Magma', 'Texturize', 'Styling Wet', 'Styling Dry', 'Styling Finish', 'Exclusiv', 'Care Brilliance', 'Care Enrich', 'Care Balance', 'Care Service', 'Care Sun', 'Smoothen', 'Hydrate', 'Repair', 'Volumize', 'Color Save', 'Shine Define', 'Clear Scalp', 'Balance Scalp', 'Expert Kit', 'Styling', 'Sun', 'Men', 'Flow', 'Form', 'Flaunt', 'Foundation', 'In Salon Service')
Quantita:int
PVendita: Double
PRivendita: Double

Associazioni

AppuntamentiClienti - Clienti:

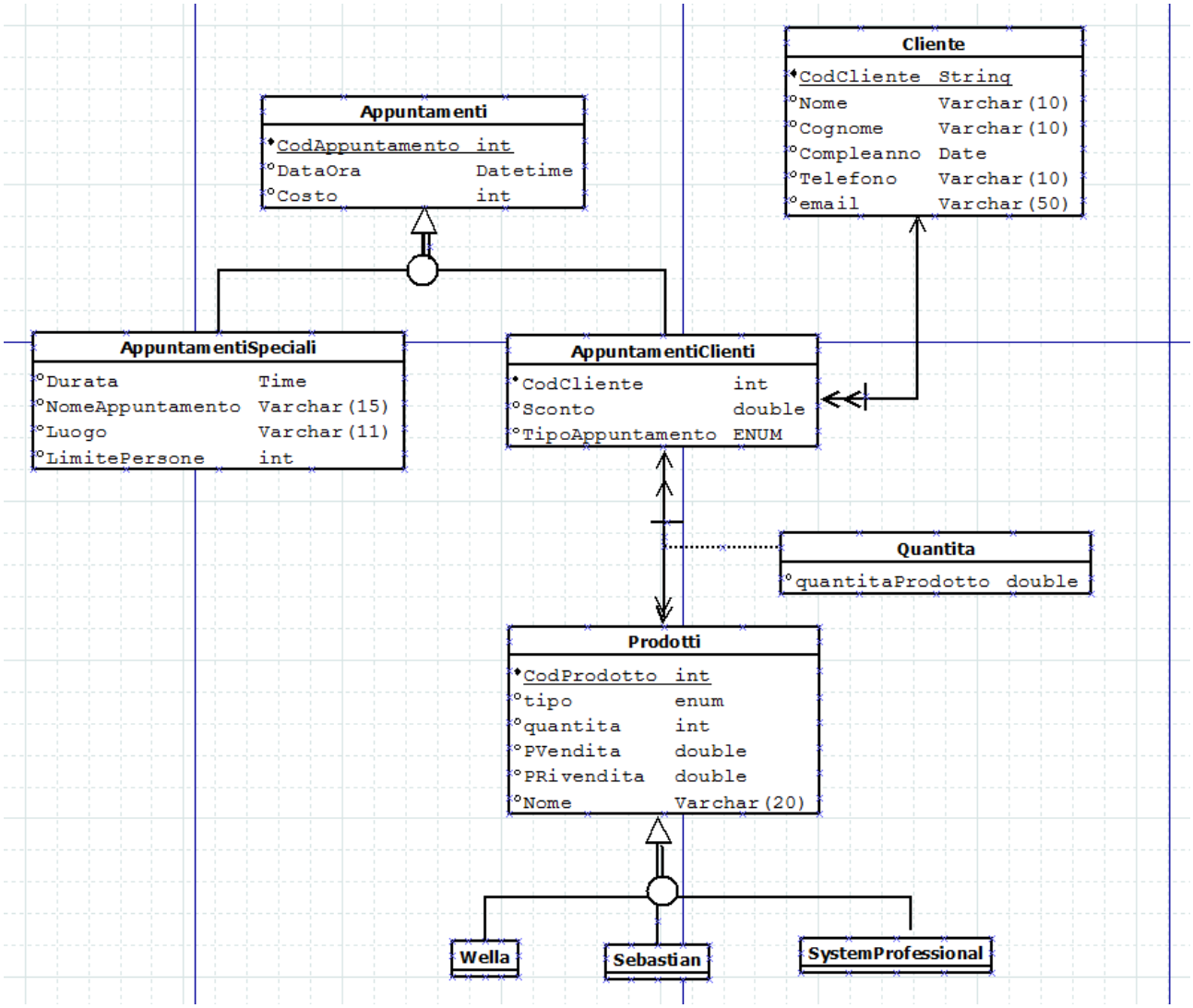
Ogni cliente ha uno o più appuntamenti (o nessuno).
Ogni AppuntamentiClienti ha un cliente
Molteplicità 1:N
Totalità:
parziale verso AppuntamentiClienti
totale verso Clienti

AppuntamentiClienti - Prodotti:

Durante un AppuntamentiClienti possono venir usati più prodotti (o nessuno). Viceversa, ogni prodotto può essere usato in più appuntamenti clienti in diverse quantità (per cliente, o anche lo stesso cliente in appuntamenti diversi).
Molteplicità: N:M
Totalità:
parziale verso Prodotti
parziale verso AppuntamentiClienti

Gerarchia delle Classi:

Gerarchia Appuntamenti: AppuntamentiSpeciali/AppuntamentiClienti
Classi Disgiunte
L'unione è una copertura


Cliente
CodCliente String
Nome Varchar(10)
Cognome Varchar(10)
Compleanno Date
Telefono Varchar(10)
email Varchar(50)
Appuntamenti
CodAppuntamento int
DataOra Datetime
Costo int
AppuntamentiSpeciali
Durata Time
NomeAppuntamento Varchar(15)
Luogo Varchar(11)
LimitePersone int
AppuntamentiClienti
CodCliente int
Sconto double
TipoAppuntamento ENUM
Quantita
quantitaProdotto double
Prodotti
CodProdotto int
tipo enum
quantita int
PVendita double
PRivendita double
Nome Varchar(20)
Wella
Sebastian
SystemProfessional

Progettazione Logica

Gerarchia delle Classi

Appuntamenti Speciali – Appuntamenti Clienti

- Partizionamento Verticale.
- Superclasse Appuntamenti (CodAppuntamento, DataOra, Costo)
- Sottoclasse AppuntamentiClienti (Sconto, TipoAppuntamento)

Appuntamento cliente può dividersi a seconda del TipoAppuntamento (discr) in unlteriori sotto classi di una relazione unica.

- Sottoclasse AppuntamentiSpeciali (Durata, NomeAppuntamento, Luogo, LimitePersone)

Appuntamenti Speciali – Appuntamenti Clienti

- Relazione Unica
- Prodotti (CodProdotto, Nome, Marca, Tipo, PVendita, PRivendita)

Marca e Tipo sono Discriminatori che producono 3 sotto classi Prodotti (Wella, SP, Sebastian) e ulteriori sottoclassi Prodotti-Wella (), Prodotti-SP (), Prodotti-Sebastian ().

Cliente:

CodCliente: int <<PK>>
Nome: VarChar(10)
Cognome: VarChar(10)
Compleanno: Date
Telefono: VarChar(10)
Email: VarChar(50)

Appuntamenti:

CodAppuntamento: int <<PK>>
DataOra: DateTime
Costo: Double

AppuntamentiSpeciali:

CodAppuntamento: int <<PK>> <FK(Appuntamenti)>
Durata: Time
NomeAppuntamento: VarChar(15)
Luogo: VarChar(20)
LimitePersone: int

AppuntamentiClienti:

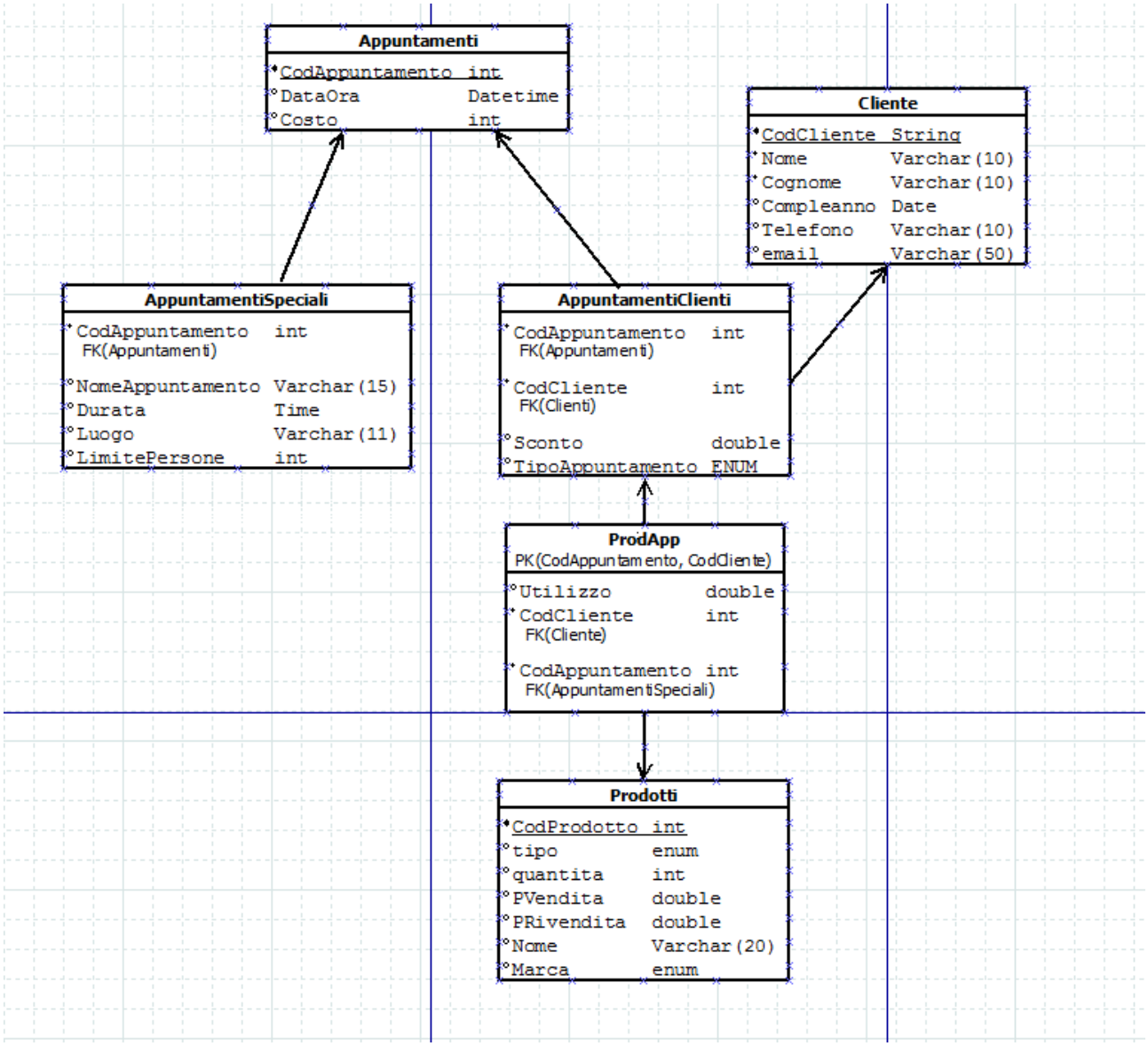
CodAppuntamento: int <<PK>><FK(Appuntamenti)>

CodCliente: int <FK(Clienti)>
Sconto: Double
TipoAppuntamento:ENUM

Prodotti:

CodProdotto: int <<PK>>
Marca: ENUM('Wella', 'SP', 'Sebastian')
Tipo: ENUM('Koleston Perfect', 'Inspire by KP', 'Welloxon Perfect', 'Color Touch', 'Color Fresh', 'Perfection', 'Eos', 'Blondor Multi-Blonde', 'Magma', 'Texturize', 'Styling Wet', 'Styling Dry', 'Styling Finish', 'Exclusiv', 'Care Brilliance', 'Care Enrich', 'Care Balance', 'Care Service', 'Care Sun', 'Smoothen', 'Hydrate', 'Repair', 'Volumize', 'Color Save', 'Shine Define', 'Clear Scalp', 'Balance Scalp', 'Expert Kit', 'Styling', 'Sun', 'Men', 'Flow', 'Form', 'Flaunt', 'Foundation', 'In Salon Service')
Quantita:int
PVendita: Double
PRivendita: Double

ProdApp:
CodProdotto: int <FK(Prodotti)>
CodCliente: int <FK(Clienti)>
Utilizzato: double


Appuntamenti
CodAppuntamento int
DataOra Datetime
Costo int
Cliente
CodCliente String
Nome Varchar(10)
Cognome Varchar(10)
Compleanno Date
Telefono Varchar(10)
email Varchar(50)
AppuntamentiSpeciali
CodAppuntamento int
FK(Appuntamenti)
NomeAppuntamento Varchar(15)
Durata Time
Luogo Varchar(11)
LimitePersone int
AppuntamentiClienti
CodAppuntamento int
FK(Appuntamenti)
CodCliente int
FK(Clienti)
Sconto double
TipoAppuntamento ENUM
ProdApp
PK(CodAppuntamento, CodCliente)
Utilizzo double
CodCliente int
FK(Cliente)
CodAppuntamento int
FK(AppuntamentiSpeciali)
Prodotti
CodProdotto int
tipo enum
quantita int
PVendita double
PRivendita double
Nome Varchar(20)
Marca enum

Implementazione Schema Logico

```
DROP TABLE IF EXISTS Clienti;

CREATE TABLE Clienti(
CodCliente INT PRIMARY KEY AUTO_INCREMENT,
Nome VARCHAR(10) NOT NULL,
Cognome VARCHAR(10) NOT NULL,
Telefono VARCHAR(10),
Email VARCHAR(50),
Compleanno DATE
)Engine=InnoDB;

INSERT INTO Clienti (CodCliente, Nome, Cognome, Telefono, Email, Compleanno) VALUES
('1','Anna Rosa', 'Cortivo', '3394188995', 'anna.cortivo@gmail.com', '1962-01-12');


DROP TABLE IF EXISTS Appuntamenti;

CREATE TABLE Appuntamenti(
CodAppuntamento INT PRIMARY KEY AUTO_INCREMENT,
DataOra DATETIME NOT NULL,
Costo DOUBLE
)Engine=InnoDB;

INSERT INTO Appuntamenti (DataOra, Costo) VALUES
('2012-06-13 10:30:00', '10'),
('2012-06-13 14:00:00', '20'),
('2012-06-14 10:30:00', '5'),
('2012-06-15 14:00:00', '50'),
('2012-06-15 10:30:00', '70'),
('2012-06-18 14:00:00', '10'),
('2012-06-20 09:00:00', '-100'),
('2012-06-21 09:00:00', '-50'),
('2012-06-22 10:00:00', '-150');


DROP TABLE IF EXISTS AppuntamentiSpeciali;

CREATE TABLE AppuntamentiSpeciali (
CodAppuntamento INT PRIMARY KEY,
Durata TIME,
NomeAppuntamento VARCHAR(15),
Luogo VARCHAR(20),
LimitePersone INT,

FOREIGN KEY(CodAppuntamento) references Appuntamenti(CodAppuntamento) on delete cascade
)Engine=InnoDB;
```

```
INSERT INTO AppuntamentiSpeciali (CodAppuntamento, Durata, NomeAppuntamento, Luogo) VALUES
('4', '02:00:00', 'Aggiornamento', 'Verona'),
('7', '01:00:00', 'AggiornamentoTaglio', 'Vicenza'),
('9', '01:30:00', 'AggiornamentoProdotti', 'Altavilla');


DROP TABLE IF EXISTS AppuntamentiClienti;

CREATE TABLE AppuntamentiClienti (
CodAppuntamento INT,
CodCliente INT,
Sconto DOUBLE,
TipoAppuntamento ENUM('shampoo','taglio','piega e phon','piega e
casco','ondulazione','colore','riflessante','decolorazione','meches','trattamenti','manicure/pedicure'),

FOREIGN KEY(CodAppuntamento) references Appuntamenti(CodAppuntamento) on delete cascade,
FOREIGN KEY(CodCliente) references Clienti(CodCliente) on delete cascade
)Engine=InnoDB;

INSERT INTO AppuntamentiClienti (CodAppuntamento, CodCliente, Sconto, TipoAppuntamento) VALUES
('1', '2', '5', 'taglio'),
('2', '3', '0', 'ondulazione'),
('3', '3', '0', 'colore'),
('4', '5', '0', 'colore'),
('5', '4', '0', 'meches'),
('6', '7', '5', 'taglio');


DROP TABLE IF EXISTS Prodotti;

CREATE TABLE Prodotti(
CodProdotto INT PRIMARY KEY,
Nome VARCHAR(20) NOT NULL,
Marca ENUM('Wella', 'SP', 'Sebastian') NOT NULL,
Tipo: ENUM('Koleston Perfect', 'Inspire by KP', 'Welloxon Perfect', 'Color Touch', 'Color Fresh', 'Perfection', 'Eos',
'Blondor Multi-Blonde', 'Magma', 'Texturize', 'Styling Wet', 'Styling Dry', 'Styling Finish', 'Exclusiv', 'Care Brilliance',
'Care Enrich', 'Care Balance', 'Care Service', 'Care Sun', 'Smoothen', 'Hydrate', 'Repair', 'Volumize', 'Color
Save', 'Shine Define', 'Clear Scalp', 'Balance Scalp', 'Expert Kit', 'Styling', 'Sun', 'Men', 'Flow', 'Form', 'Flaunt',
'Foundation', 'In Salon Service') NOT NULL,
Quantita INT NOT NULL,
PVendita DOUBLE,
PRivendita DOUBLE
)Engine=InnoDB;


DROP TABLE IF EXISTS ProdApp;

CREATE TABLE ProdApp(
CodAppuntamento INT,
CodProdotto INT,
Utilizzo DOUBLE,
```

```
FOREIGN KEY(CodProdotto) REFERENCES Prodotti(CodProdotto) ON UPDATE CASCADE,
FOREIGN KEY(CodAppuntamento) REFERENCES AppuntamentiClienti(CodAppuntamento) ON UPDATE
CASCADE ON DELETE CASCADE
)Engine=InnoDB;
```

La gestione degli errori da parte dei trigger sono state implementate grazie una classe Eccezioni così descritta

```
DROP TABLE IF EXISTS Eccezioni;

CREATE TABLE Eccezioni(
  Id_Eccezione INT PRIMARY KEY,
  Descrizione VARCHAR(50)
)Engine=InnoDB;

INSERT INTO Eccezioni(Id_Eccezione, Descrizione) VALUES
('1', 'Errore 1: Errore di Inserimento, e presente un appuntamento speciale'),
('2', 'Errore 2: Errore di Inserimento, il prodotto che si vuole inserire e gia inserito');
```

QUERY

1. Query che ritorna i clienti che compiono gli anni da qui a un mese

```
SELECT *
FROM Clienti
WHERE Compleanno BETWEEN CURDATE() AND (ADDDATE(CURDATE(), INTERVAL 31 DAY));

In questa Query si selezionano tutti e soli I clienti che hanno il compleanno
dalla data corrente a un mese.

OUTPUT:
+------------+-------+---------+----------+-------+------------+
| CodCliente | Nome  | Cognome | Telefono | Email | Compleanno |
+------------+-------+---------+----------+-------+------------+
|         11 | Maria | Luisa   |          | @     | 2012-07-29 |
+------------+-------+---------+----------+-------+------------+
1 row in set (0.00 sec)
```

2. Query che ritorna tutti i prodotti in esaurimento (cioè con quantità minore di 4) senza considerare i prodotti già esauriti, che sono quelli che non vengono abitualmente ordinati dal catalogo.

```
SELECT *
FROM Prodotti
WHERE quantita<4 AND quantita!=0

In questa query vengono ritornati tutti i prodotti i quali in magazzino ci
sono meno di 4 confezioni, ma non considera i prodotti che non ci sono in
magazzino, perché son quelli che non vengono scelti dal catalogo abitualmente

OUTPUT:
+-------------+----------------------+-----------+------------------+----------+
| CodProdotto | Nome                 | Marca     | Tipo             | Quantita |
+-------------+----------------------+-----------+------------------+----------+
|           5 | Smoothen Mask 200 ml | SP        | Smoothen         |        2 |
|           6 | Smoothen Mask 400 ml | SP        | Smoothen         |        3 |
|           7 | Smoothen Infusion 5  | SP        | Smoothen         |        2 |
|          12 | Hydrate Mask 200 ml  | SP        | Hydrate          |        1 |
|          14 | Hydrate Emulsion 50  | SP        | Hydrate          |        3 |
|          24 | Repair Infusion 5 ml | SP        | Repair           |        2 |
|          33 | Volumize Infusion 5  | SP        | Volumize         |        2 |
|          79 | After Sun Fluid 125  | SP        | Sun              |        1 |
|          96 | Precise Shine 75 ml  | SP        | Men              |        2 |
|         145 | Penetraitt Condition | Sebastian | In Salon Service |        2 |
+-------------+----------------------+-----------+------------------+----------+
10 rows in set (0.00 sec)
```

3. Query che ritorna la classifica dei primi dieci prodotti più usati.

```
SELECT p.Nome, a.Utilizzo
```

```
FROM Prodotti p NATURAL JOIN ProdApp a
ORDER BY Utilizzo DESC
LIMIT 10;
```

In questa query vengono ordinati in modo decrescente i prodotti in base al loro utilizzo durante gli appuntamenti con i clienti, e tra questi vengono tornati i primi 10, cioè i più usati.

```
+----------------------+----------+
| Nome                 | Utilizzo |
+----------------------+----------+
| Hydre Conditioner 10 |      1.6 |
| Light Conditioner 10 |      1.3 |
| Light Conditioner 10 |      0.2 |
+----------------------+----------+
3 rows in set (0.01 sec)
```

4. Questa query torna lo "storico" dei prodotti usati per un cliente, cioè l'elenco dei prodotti con le relative quantità usate durante gli appuntamenti dei clienti. Il Cliente dell'esempio è quello con CodCliente='5'

```
CREATE VIEW storico AS
SELECT a.CodAppuntamento, a.DataOra, pa.CodProdotto, pa.Utilizzo
FROM AppuntamentiClienti a NATURAL JOIN ProdApp pa;

SELECT s.Codappuntamento, s.DataOra, s.CodProdotto, s.Utilizzo, p.Nome
FROM storico s NATUAL JOIN Prodotto p
WHERE CodCliente = 2;
```

Nella Vista creata vengono considerati tutti gli appuntamenti durante i quali si è usata una certa quantità di prodotto, e di questi tiene nota del codice dell'appuntamento, la data e l'ora, il codice del prodotto un questione e quant'è l'utilizzo in millilitri.
Nella select viene scelto il cliente con codice 2, e di questo riturnano tutte le colonne della View storico ed in più il nome del prodotto di cui è riportata la quantità utilizzata (per una più facile lettura).

OUTPUT:

```
+-----------------+---------------------+-------------+----------+----------------------+
| CodAppuntamento | DataOra             | CodProdotto | Utilizzo | Nome                 |
+-----------------+---------------------+-------------+----------+----------------------+
|              11 | 2012-07-12 13:00:00 |         144 |      1.3 | Light Conditioner 10 |
|              11 | 2012-07-12 13:00:00 |         144 |      0.2 | Light Conditioner 10 |
|              11 | 2012-07-12 13:00:00 |         143 |      1.6 | Hydre Conditioner 10 |
+-----------------+---------------------+-------------+----------+----------------------+
3 rows in set (0.00 sec)
```

5. Query che ritorna le percentuali di frequenza dei vari tipi di appuntamento richiesti dai clienti

```
CREATE VIEW Contatori(Parziali, Tipo) AS
```

```
SELECT COUNT(*), TipoAppuntamento
FROM AppuntamentiClienti
GROUP BY TipoAppuntamento;

SELECT (p.Parziali/COUNT(*))*100 AS percentuale, a.TipoAppuntamento
FROM Contatori p NATURAL JOIN AppuntamentiClienti a
GROUP BY a.TipoAppuntamento;
```

Nella View creata si contano i tipi di appuntamenti per gli appuntamenti con i clienti. Nella select invece tornano le percentuali dei tipi d appuntamenti

OUTPUT:

```
+--------------------+--------------------------------+
|   percentuale      |        TipoAppuntamento        |
+--------------------+--------------------------------+
|    100.0000        |              meches            |
+--------------------+--------------------------------+
1 row in set (0.00 sec)
```

6. Query che ritorna il tipo do appuntamento più usato tra chi ha più appuntamenti

```
CREATE View Fedele(fedele) AS
SELECT a.CodCliente, c.Nome, c.Cognome
FROM AppuntamentiClienti NATURAL JOIN Clienti
GROUP BY CodCliente
ORDER BY COUNT(*) DESC
LIMIT 1


SELECT MAX(c.Parziali), a.CodCliente, a.TipoAppuntamento
FROM AppuntamentiClienti a NATURAL JOIN Contatori c JOIN Fedele f
ON(a.CodCliente= f.fedele)
```

```
+-----------------+------------+------------------+
| MAX(c.Parziali) | CodCliente | TipoAppuntamento |
+-----------------+------------+------------------+
|               1 |          2 | meches           |
+-----------------+------------+------------------+
1 row in set (0.00 sec)
```

TRIGGER

1. Il primo trigger controlla il problema di un inserimento Appuntamento in contemporanea ad un appuntamento Speciale. Quindi prima di inserire un appuntamento, il database, controlla che non si accavalli con un appuntamento di tipo speciale.

```
DROP TRIGGER IF EXISTS ControlloAppuntamentiSpeciali;
delimiter $$
CREATE TRIGGER ControlloAppuntamentiSpeciali BEFORE INSERT ON
Appuntamenti

FOR EACH ROW
BEGIN
	DECLARE AppSpeciali INT;

	SELECT count(*) INTO AppSpeciali FROM AppuntamentiSpeciali a JOIN
Appuntamenti b
	WHERE b.DataOra<NEW.DataOra AND
NEW.DataOra<(ADDTIME(b.DataOra,a.Durata));

	IF(AppSpeciali) THEN
		SELECT Descrizione FROM Eccezione WHERE Id_Eccezione=1;
		INSERT INTO Eccezioni(Id_Eccezione) VALUES ('1');
	END IF;
END$$
delimiter ;
```

2. Il secondo trigger controlla che non si possa inserire un prodotto se non è terminato, cioè non posso aumentare (inserire l'ordine) una quantità di prodotto se non abbiamo

```
DROP TRIGGER IF EXISTS InserimentoProdotto;
delimiter $$
CREATE TRIGGER InserimentoProfo BEFORE INSERT ON Prodotti
FOR EACH ROW
BEGIN
	DECLARE presente INT;


	SELECT count(*) INTO presente
	FROM Prodotti
	WHERE quantita>1

	IF(!presente) THEN
		SELECT Descrizione FROM Eccezione WHERE Id_Eccezione=2;
		INSERT INTO Eccezioni(Id_Eccezione) VALUES ('2');
	END IF;

END $$
```

FUNZIONI

1. La funzione torna il guadagno del mese, inteso come somma algebrica tra i costi degli appuntamenti (considerando entrate i costi degli appuntamenti Clienti e uscite quelli Speciali) che sono avvenuti/avvengono questo mese.

```
DELIMITER $
CREATE FUNCTION Guadagno()
RETURNS INT
BEGIN
	DECLARE guadagno INT;

	SELECT SUM(Costo) INTO guadagno
	FROM Appuntamenti
	WHERE MONTH(DataOra)=MONTH(CURDATE());
	RETURN guadagno;
END $
```

2. La funzione ritorna il tipo di Appuntamento richiesto tra gli AppuntamentiClienti.

```
DELIMITER $
CREATE FUNCTION TipoPreferito(aNome VARCHAR(10), aCognome VARCHAR(10))
RETURNS ENUM('shampoo','taglio','piega e phon','piega e
casco','ondulazione','colore','riflessante','decolorazione','meches','tr
attamenti','manicure/pedicure')
BEGIN
	DECLARE CodCli INT;
	DECLARE Preferenze ENUM('shampoo','taglio','piega e phon','piega
e
casco','ondulazione','colore','riflessante','decolorazione','meches','tr
attamenti','manicure/pedicure');

	SELECT CodCliente INTO CodCli
	FROM Cliente
	WHERE Nome=aNome AND Cognome=aCognome;

	SELECT a.TipoAppuntamento INTO Preferenze
	FROM AppuntamentiClienti a
	WHERE a.CodCliente=CodCli
	GROUP BY TipoAppuntamento
	ORDER BY COUNT(*) DESC
	LIMIT 1;

	RETURN Preferenze;
END $
```

Procedure

1. La procedura aggiorna i campi di un prodotto, con le modifiche a essa passata.

```
DELIMITER $
CREATE PROCEDURE AggiornamentoProdotti
(
IN aCodProd INT,
IN aQuantita INT,
IN aPVendita DOUBLE,
IN aPRivendita DOUBLE)

BEGIN

UPDATE Prodotti
SET Quantita=aQuantita AND PVendita=aPVendita AND PRivendita0aPRivendita
WHERE CodProd=aCodProd

END $
DELIMITER ;
```

2. La procedura inserisce un nuovo appuntamento cliente

```
DELIMITER $
CREATE PROCEDURE InserimentoAppuntamentoCliente
(
IN aCodCliente INT,
IN aDataOra DATETIME,
IN aSconto DOUBLE,
IN aCosto DOUBLE,
IN aTipo ENUM ('shampoo', 'taglio', 'piega e phon', 'piega e casco', 'ondulazione', 'colore', 'riflessante',
'decolorazione', 'meches', 'trattamenti', 'manicure/pedicure'))

BEGIN
DECLARE CodiceApp INT;
INSERT INTO Appuntamenti(DataOra, Costo) VALUES (ADataOra, aCosto);

SELECT CodAppuntamento INTO CodiceApp FROM Appuntamenti WHERE DataOra=aDataOra AND
Costo=aCosto;

INSERT INTO AppuntamentiClienti(CodAppuntamento, CodCliente, Sconto, TipoAppuntamento) VALUES
(CodiceApp, aCodCliente, aSconto, aTipo);

END $
DELIMITER ;
```

3. Dato un prodotto (codice prodotto) la procedura lo elimina, cioè azzera la sua quantità

```
DELIMITER $
CREATE PROCEDURE EliminaProdotto
(IN aCodProdotto INT)

BEGIN
```

```
UPDATE Prodotti SET Quantita=0 WHERE CodProdotto=aCodProdotto;

END $
DELIMITER ;
```

4. La procedura inserisce un nuovo appuntamento di tipo speciale

```
DELIMITER $
CREATE PROCEDURE InserimentoAppuntamentoSpeciale
(
IN aDataOra DATETIME,
IN aCosto DOUBLE,
IN aDurata TIME,
IN aNomeApp VARCHAR(15),
IN aLuogo VARCHAR(20),
IN aLimPersone INT)

BEGIN
DECLARE CodiceApp INT;
INSERT INTO Appuntamenti(DataOra, Costo) VALUES (ADataOra, aCosto);

SELECT CodAppuntamento INTO CodiceApp FROM Appuntamenti WHERE DataOra=aDataOra AND
Costo=aCosto;

INSERT INTO AppuntamentiSpeciali(CodAppuntamento, Durata, NomeApp, Luogo, LimitePersone) VALUES
(CodiceApp, aDurata, aNomeApp, aLuogo, aLimPersone);

END $
DELIMITER ;
```

5. La procedura inserisce un nuovo Cliente nella tabella cliente.

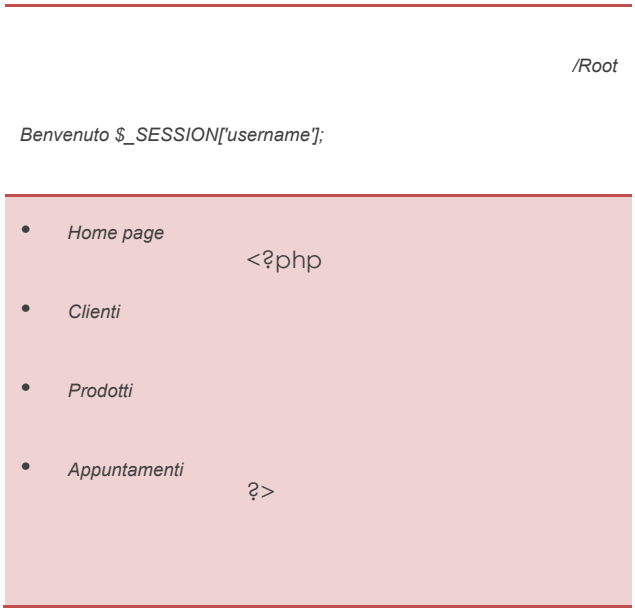
```
DELIMITER $
CREATE PROCEDURE InserimentoNewCliente
(
IN aNome VARCHAR(10),
IN aCognome VARCHAR(10),
IN aTelefono VARCHAR(10),
IN aEmail VARCHAR(50),
IN aCompleanno DATE)

BEGIN
INSERT INTO Clienti(Nome, Cognome, Telefono, Email, Compleanno) VALUES
(aNome, aCognome, aTelefono, aEmail, aCompleanno);

END $
DELIMITER ;
```

Architettura dell'Applicazione Web

Interfaccia

L'interfaccia con cui si sposa la base di dati è organizzata da una tabella (senza bordi) formata in codice html.

La parte sinistra di ogni pagine è un portale alle principali classi gestite nel progetto, mentre la parte sulla destra contiene il corpo delle nostre funzioni, pagine e tabelle.
In alto con allineamento sinistro il sito mostra la pagina in cui ci si trova. Infine in alcune zone di particolare spessore il server ti avvisa di essere loggato correttamente.



Funzionalità Relative e Scelte Procedurali

Clienti :

- *Compleanni Questo mese :* Query che ritorna i clienti che compiono gli anni da qui a un mese
- *Nuovo Cliente :* Pagina per aggiungere un nuovo cliente al database, c'è la possibilità di inserire l'appuntamento subito dopo passsando con metodo post (hidden) già il CodCliente appena creato
- *Storico Prodotti :* torna lo "storico" dei prodotti usati per un cliente, cioè l'elenco dei prodotti con le relative quantità usate durante gli appuntamenti dei clienti. Il Cliente dell'esempio è quello con CodCliente='5'

Prodotti :

- *Prodotti in Esaurimento :* Query che ritorna i prodotti con un intero minore di 5 in quantità
- *Maggior Numero Prodotti Usati :* classifica dei primi dieci prodotti più usati.

- *Gestione Prodotti : possibilità dinamica di modificare la quantità di ogni prodotto, se impostate quantita negative il prodotto viene cancellato*

Appuntamenti :

- *Nuovo Appuntamento :* Da la possibilità di inserire un nuovo appuntamento
- *Tipo Appuntamento Frequente :* Ritorna la query le percentuali di frequenza dei vari tipi di appuntamento richiesti dai clienti
- *Ricerca Appuntamenti : Colonna portante dell'applicazione web. In cui puoi ricercare appuntamenti e prodotti. Con metodo, anche incrociato , per data e per cliente. Trovato l'appuntamento desiderato è completamente modificabile, cancellabile ed è infine possibile aggiungere quantità e nuovo prodotto.*

Tipo di Autenticazione

L'autenticazione mantiene lo stato dell'applicazione web in pagine distinte utilizzando le sessioni.

Alla prima richieste da parte del browser di una risorsa, viene creata una sessione con un session id(sid), ritornato al client. Nelle richieste successive, il browser fornisce il session id per identificare la sessione (l'utente è connesso); quindi il server può recuperare le variabili sessione relative.

Una pagina non controllata dagli utenti dell'applicazione conterrà una verifica alla base di dati dove vi sono riportate in tabella nome utente e password delle persone registrate. Nel caso il login e/o chiave per poter entrare venissero sbagliati automaticamente non si potrà accedere alla pagina desiderata e riportati alla pagina di login.