Ringraziamenti

Desidero ringraziare la prof.ssa Bergamaschi, relatore di questa tesi, per la grande disponibilità e cortesia dimostratemi, e per tutto l'aiuto fornito durante la stesura.

Un sentito ringraziamento ai miei genitori, che, con il loro incrollabile sostegno morale ed economico, mi hanno permesso di raggiungere questo traguardo.

Desidero inoltre ringraziare la ditta A.E.B. srl, tutti i ragazzi dello staff di ricerca e sviluppo, e in particolare Claudio, per tutto quanto hanno fatto per me durante il periodo di stage.

Un ultimo ringraziamento ai compagni di studi, per essermi stati vicini sia nei momenti difficili, sia nei momenti felici: sono stati per me più veri amici che semplici compagni.

INDICE GENERALE

INTRODUZIONE

La diffusione dei sistemi informatici nelle aziende ha raggiunto un livello tale da vedere impiegate soluzioni software per automatizzare anche le attività più semplici. Le applicazioni software non solo sono in grado di velocizzare il lavoro manuale, ma possono anche semplificare operazioni complesse. In una realtà come quella odierna, è normale impiegare software applicativo anche per esigenze semplici, come la gestione di un piccolo magazzino di componenti elettronici.
È quanto mi è stato chiesto di realizzare dall'azienda AEB s.r.l., presso cui ho svolto l'attività di stage universitario di tre mesi.

AEB progetta e produce da anni prodotti elettronici ed elettroacustici nel settore audio professionale, utilizzando il marchio "dB Technologies". Il reparto produzione dispone di un magazzino che, negli anni, ha assunto dimensioni sempre più elevate, imponendo una gestione economica e burocratica di una certa complessità. In questa situazione i laboratori di ricerca e sviluppo dell'azienda erano costretti a servirsi del magazzino tramite tutte le procedure necessarie, con i conseguenti tempi di attesa. Per ridurre i disagi dovuti ai continui spostamenti alla ricerca di componenti, l'azienda ha deciso di creare un ulteriore zona adibita a magazzino, separata da quello principale, di dimensioni molto contenute, ad esclusivo utilizzo dei laboratori di ricerca e sviluppo.

La presente tesi è il risultato del lavoro svolto per la realizzazione dell'applicazione "Magazzino dB". Oltre a descrivere nel dettaglio il software creato, gli studi compiuti, e le scelte fatte durante il progetto, è stato dedicato un capitolo per descrivere più specificamente il codice del programma: in questo modo il presente elaborato può essere visto anche come manuale di riferimento dedicato a chiunque si volesse accingere, in futuro, a modificare o ampliare l'applicazione attuale.

Verrà ora esposta la struttura secondo la quale è organizzata questa tesi.

Nel primo capitolo è descritta in modo dettagliato ogni funzionalità dell'applicazione, accompagnata da immagini che ne chiariscono l'utilizzo e l'interfaccia grafica.
Nel secondo capitolo si procede alla descrizione dei sistemi utilizzati per la realizzazione del software, ai motivi di tali scelte e ad una breve analisi delle soluzioni concorrenti. In particolare sono descritti nel dettaglio il sistema MySQL, e le Active Server Pages (ASP).
Nel terzo capitolo è trattata la parte più tecnica, a partire dalla progettazione concettuale (mediante gli schemi E/R), per arrivare alla parte di programmazione vera e propria, con la descrizione delle funzioni realizzate con codice ASP. Gli esempi di

codice sono preceduti da un'introduzione sul funzionamento di alcuni aspetti di ASP, quali le sessioni e l'accesso ai sistemi DBMS.
Il quarto capitolo è dedicato alle conclusioni, dove sono descritti alcune difficoltà e problemi riscontrati durante lo sviluppo. Nello stesso capitolo sono date alcune indicazioni riguardo i possibili sviluppi futuri del programma.
Chiudono la tesi la bibliografia ed un breve glossario dei termini tecnici utilizzati nel corso della trattazione.

1. PANORAMICA DELL'APPLICAZIONE

1.1 Specifica dei requisiti del software

L'applicazione è stata creata al fine di facilitare e velocizzare la gestione automatizzata di un piccolo magazzino di elementi elettronici, elettrici e acustici.

Al momento di avviare il progetto sono state raccolte con l'azienda le specifiche dei requisiti del software, ovvero le funzioni di cui l'azienda aveva bisogno. A seguire vengono riportate nel dettaglio le principali funzioni per la gestione del magazzino:

- Inserimento, modifica ed eliminazione degli articoli, e gestione della collocazione in magazzino;
- Prelievo di quantità di prodotti, e relativa gestione delle quantità rimanenti;
- Gestione dei prodotti a fine scorta e approvvigionamento;
- Visualizzazione e stampa del totale dei prelievi effettuati in una sessione;

I requisiti dell'applicazione comprendono inoltre:

- Ricerca nel database veloce e flessibile, in grado di gestire un numero variabile di parametri di ricerca indicati dall'utente;
- Funzionamento sulla rete intranet dell'azienda;
- Gestione della sicurezza con accesso mediante nome utente e password;
- Gestione degli utenti che hanno accesso al sistema;
- Massima semplicità d'utilizzo.

1.2 Descrizione generale

L'obiettivo principale che è stato posto durante il progetto è stato quello di creare un'interfaccia quanto più semplice e immediata possibile.

Il software "Magazzino Laboratorio dB", così è stato chiamato, non è infatti destinato a magazzinieri o professionisti di logistica, ma a tecnici elettronici ed ingegneri, che dovranno utilizzarlo saltuariamente e senza aver bisogno di funzionalità troppo complesse.

L'interfaccia si presenta come un semplice sito web, con la pagina divisa in due parti: una sezione superiore contenente il menu a disposizione dell'utente, e una inferiore, la principale, dove compaiono tutte le pagine di ricerca e le tabelle dei risultati richiesti.

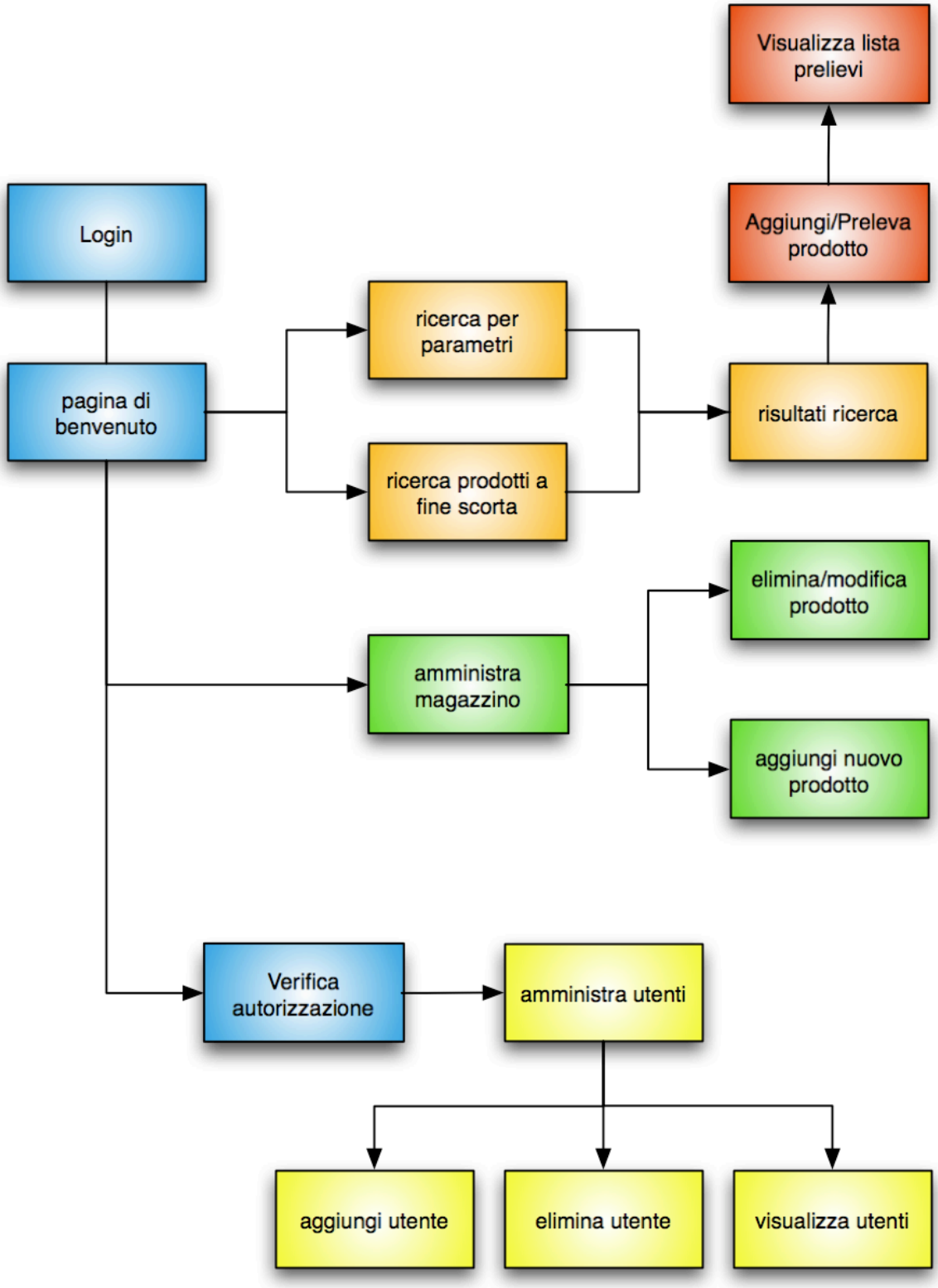
Per perseguire l'obiettivo di mantenere la massima semplicità d'uso, il menu presenta solo cinque opzioni generiche, ognuna delle quali rimanda alla rispettiva sottosezione, dove si trovano le funzioni specifiche.

L'uso di un'interfaccia web ha permesso un'ulteriore semplificazione: dopo aver installato la prima volta il software sul server, nessuna ulteriore installazione è richiesta all'utente: ogni postazione collegata all'intranet aziendale è in grado di accedere all'applicazione, tramite l'utilizzo di un semplice browser web, presente nella dotazione standard di tutti gli attuali sistemi operativi.

Nella pagina seguente è riportata la mappa del sito creato, dove sono riportate tutte le schermate relative alle funzioni e i collegamenti fra una sezione e l'altra.

Nei paragrafi a seguire sono invece descritti nel dettaglio il funzionamento e le schermate delle singole funzioni.

Mappa del sito creato



1.3 Descrizione delle singole funzioni

Login

La pagina di login è la prima che si presenta all'utente all'avvio dell'applicazione; questa consente di proteggere l'accesso ai dati da parte degli utenti non autorizzati.

figura 1.1 - pagina di login

I nuovi utenti del sistema, e i relativi UserID e Password, sono inseribili solo da un altro utente che ha privilegi di amministratore. Durante l'immissione dei dati, lo User ID viene visualizzato in chiaro, mentre la password viene nascosta da asterischi per motivi di sicurezza. Al termine dell'immissione, il programma si collega al database e verifica la correttezza dei dati immessi: nel caso in cui l'utente risultasse inesistente o la password errata, il programma tornerà semplicemente a mostrare la pagina del login.

Amministrazione utenti

In questa sezione un utente dotato di diritti di amministrazione può visualizzare l'elenco degli utenti che hanno accesso al sistema, aggiungerne nuovi o eliminare utenti esistenti.

Il processo di inserimento di un nuovo utente consiste nello scegliere uno UserID e una Password, e come dati aggiuntivi vengono richiesti nome e cognome completi dell'utente da aggiungere.

La procedura di eliminazione prevede, per evitare cancellazioni non volute, di riportare nome e cognome dell'utente che si vuole eliminare dal sistema.

Prima di eliminare un utente, è necessario che l'operatore esegua una nuova autenticazione come amministratore.

Figura 1.2 - inserimento un nuovo utente

Inserimento articoli

Nel magazzino sono presenti cinque grandi categorie di prodotti: elettronici, acustici, meccanici, elettrici ed elementi attivi. E' stata fatta una divisione in categorie perché le caratteristiche dei prodotti sono troppo differenti perché possano essere descritte allo stesso modo.

La maschera di inserimento è suddivisa dunque in cinque sezioni, una per ogni categoria. É richiesto all'utente di inserire le varie caratteristiche, ma non è obbligatorio completare ogni campo: solo i campi "tipo" e "codice" sono strettamente necessari (in quanto vengono memorizzati come primary key nel database).

Al termine dell'inserimento un tasto "inserisci" lancia la query di inserimento al database, mentre un tasto "reset" azzera tutti i campi inseriti fino a quel momento.

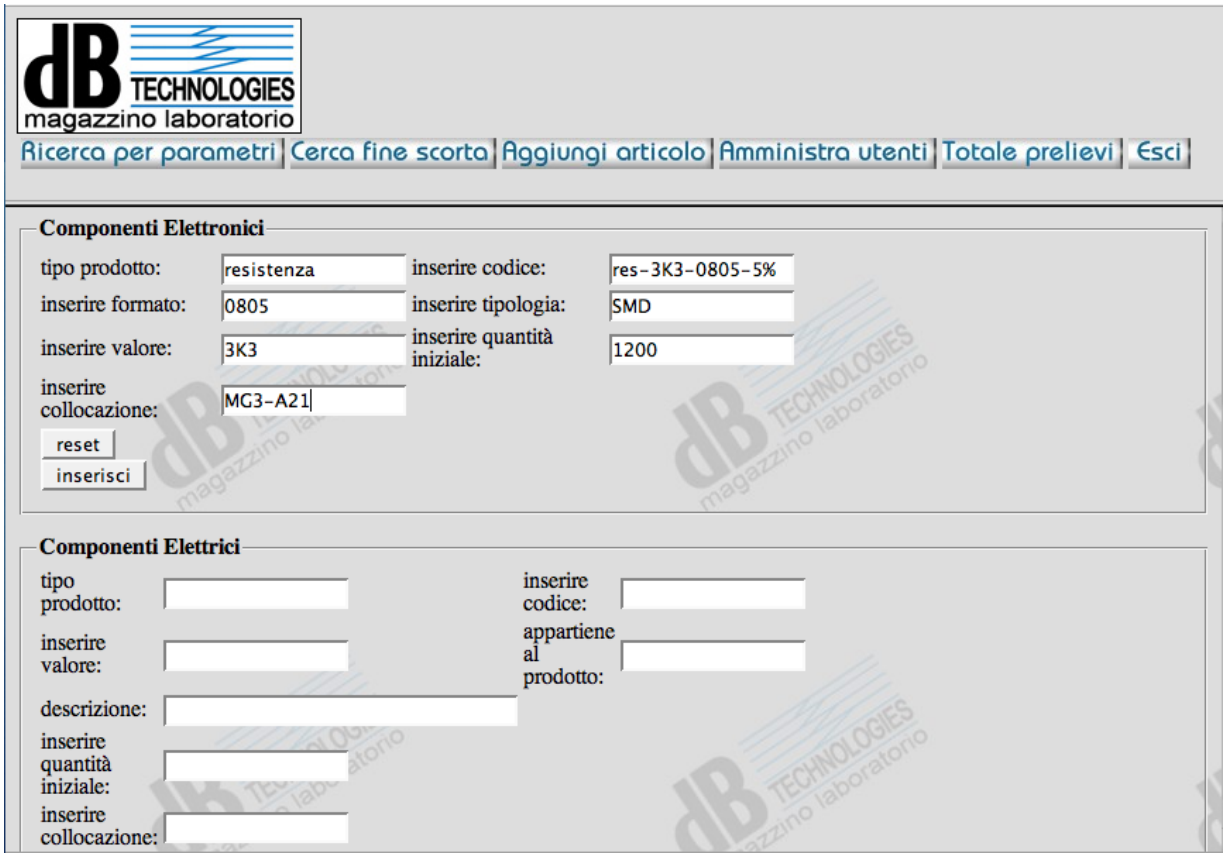


Figura 1.3 - inserimento di un nuovo articolo

Ricerca articoli

La sezione di ricerca articoli è, come la pagina d'inserimento, divisa in cinque sezioni, una per ogni categoria di prodotti contenuti nel magazzino.

La funzione di ricerca accetta un numero variabile di parametri, in modo che l'utente possa portare a termine una ricerca conoscendo anche solo una delle caratteristiche del prodotto cercato.

Una query al database, eseguita all'apertura della pagina, permette di creare un menu a tendina per selezionare la tipologia del prodotto fra quelle effettivamente presenti nel database.

I risultati della ricerca sono visualizzati mediante una tabella; se la ricerca non ha portato risultati l'utente sarà avvisato e gli sarà proposto di tornare alla pagina di ricerca.

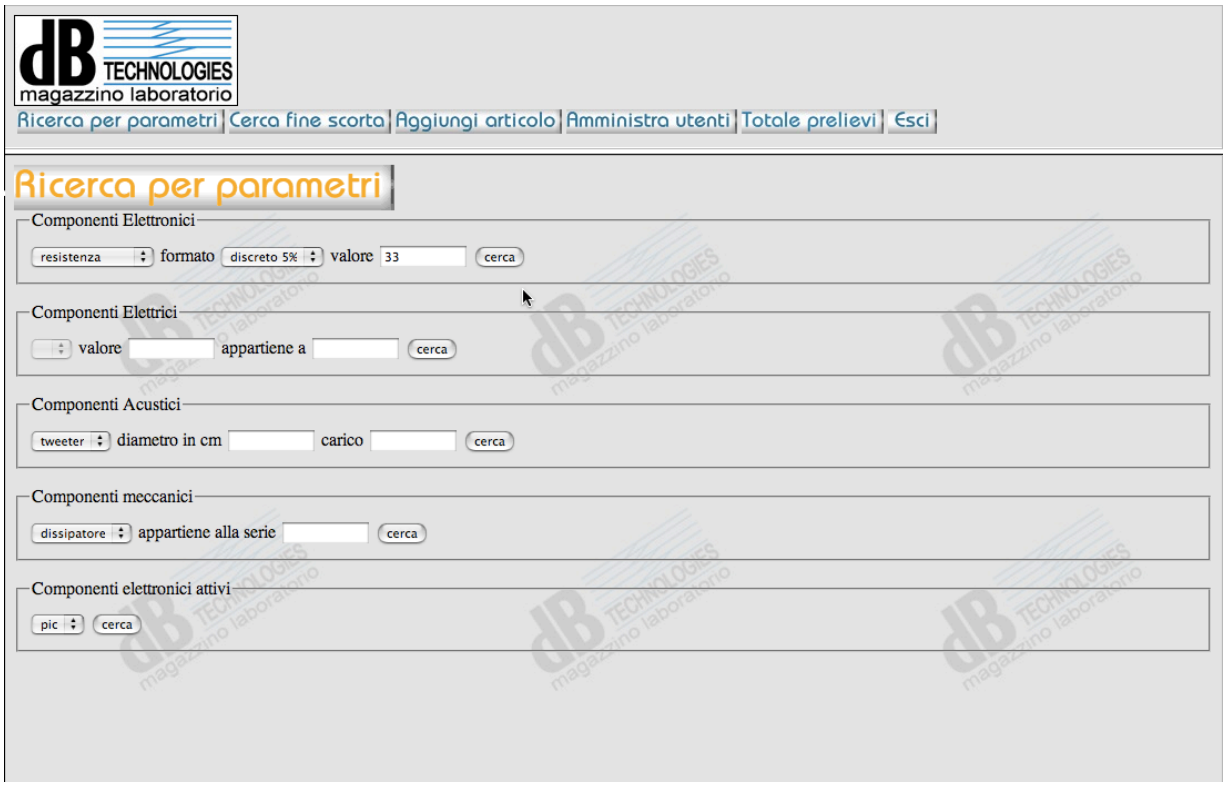


Figura 1.4 - ricerca per parametri

Modifica, quantità e rimozione di un articolo

Una volta effettuata una ricerca e visualizzati i risultati, l'applicazione propone all'utente di selezionare un articolo fra quelli trovati e di eseguire un'azione sull'articolo interessato.

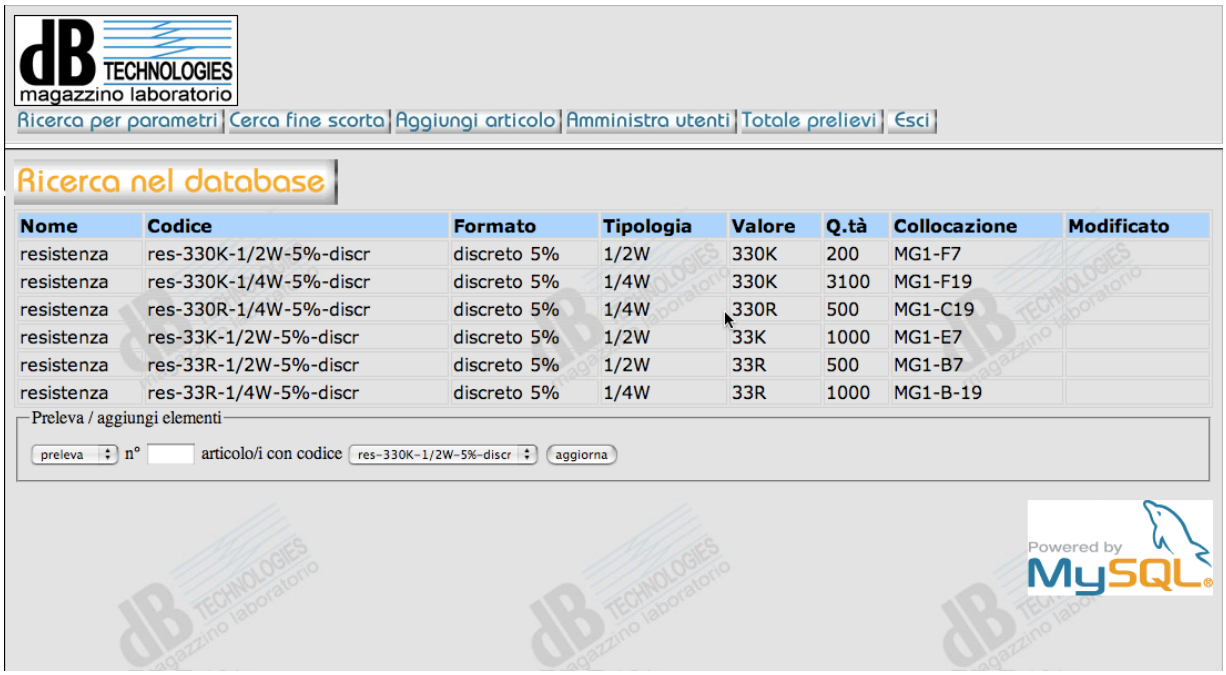


Figura 1.5 - risultati della ricerca e opzioni disponibili

Le azioni possibili sono quattro:

- Aggiungi: questo serve ad aggiungere una certa quantità del prodotto selezionato, utile quando si è provveduto al riordino della merce a fine scorta.
- Preleva: questa funzione permette di prelevare un numero di elementi del prodotto selezionato. Una query al database controlla che la quantità richiesta sia inferiore o uguale a quella disponibile in magazzino; In caso contrario il prodotto non sarà prelevato, e l'utente sarà avvisato della disponibilità insufficiente del prodotto.
- Modifica: tramite questa opzione l'utente può modificare tutte le caratteristiche del prodotto selezionato, per esempio per rimuovere errori commessi durante l'immissione o per inserire nuove descrizioni che possano aiutare nella ricerca.
- Rimuovi: selezionando questa funzione si chiede al sistema di rimuovere completamente un prodotto dal database; vista l'irreversibilità dell'azione, l'applicazione chiederà all'utente di confermare la propria scelta.

Lista dei prelievi

Durante l'uso dell'applicazione, l'utente può in qualunque momento visualizzare una lista dei prodotti che ha prelevato, in relazione alla sessione corrente. La lista comprende il nome del prodotto, il relativo codice e la quantità prelevata.

Una volta terminata la sessione (mediante la pressione del tasto "esci"), la lista è azzerata, ed è pronta a memorizzare un nuovo prelievo.

L'applicazione permette di stampare la pagina in modo da avere un resoconto scritto delle operazioni eseguite.

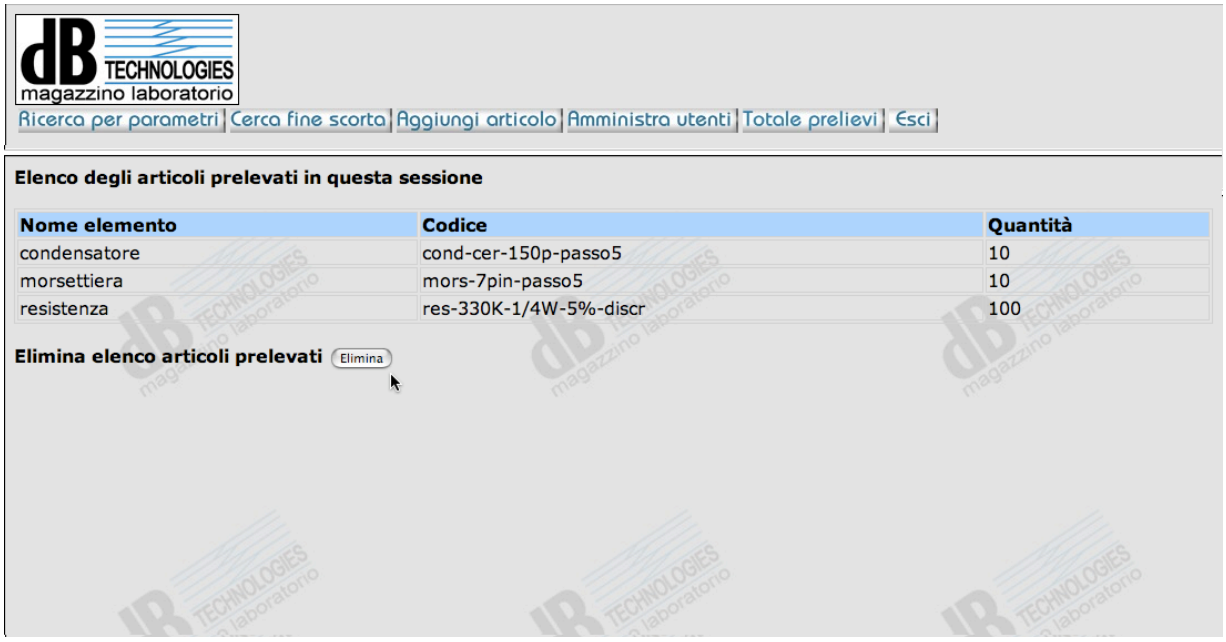


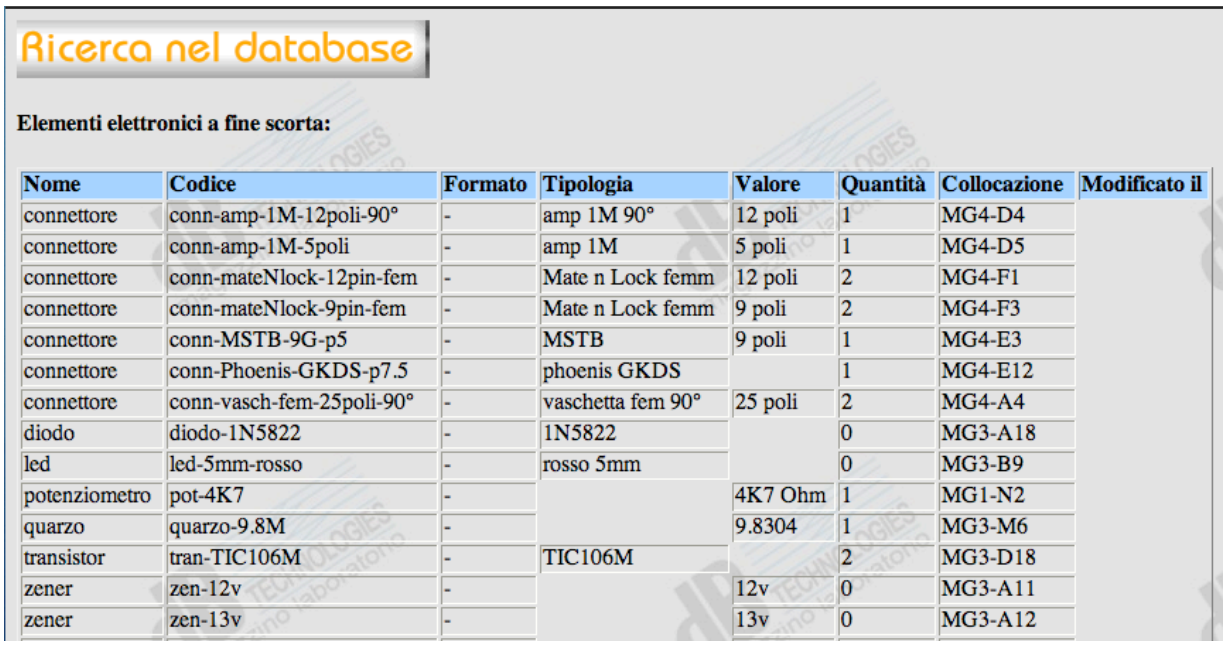
Figura 1.6 - lista dei prelievi

Ricerca articoli a fine scorta

Selezionando il pulsante “Cerca fine scorta” si accede alla funzione che permette di individuare gli articoli che sono presenti nel magazzino in una quantità inferiore ad un numero indicato.

La pagina chiama una query SQL, una per ogni categoria di prodotti, e cerca gli articoli a fine scorta. I risultati sono visualizzati in una tabella stampabile.

Figura 1.7 - impostazione del limite di fine scorta



Nome	Codice	Formato	Tipologia	Valore	Quantità	Collocazione	Modificato il
connettore	conn-amp-1M-12poli-90°	-	amp 1M 90°	12 poli	1	MG4-D4	
connettore	conn-amp-1M-5poli	-	amp 1M	5 poli	1	MG4-D5	
connettore	conn-mateNlock-12pin-fem	-	Mate n Lock femm	12 poli	2	MG4-F1	
connettore	conn-mateNlock-9pin-fem	-	Mate n Lock femm	9 poli	2	MG4-F3	
connettore	conn-MSTB-9G-p5	-	MSTB	9 poli	1	MG4-E3	
connettore	conn-Phoenis-GKDS-p7.5	-	phoenis GKDS		1	MG4-E12	
connettore	conn-vasch-fem-25poli-90°	-	vaschetta fem 90°	25 poli	2	MG4-A4	
diodo	diodo-1N5822	-	1N5822		0	MG3-A18	
led	led-5mm-rosso	-	rosso 5mm		0	MG3-B9	
potenziometro	pot-4K7	-		4K7 Ohm	1	MG1-N2	
quarzo	quarzo-9.8M	-		9.8304	1	MG3-M6	
transistor	tran-TIC106M	-	TIC106M		2	MG3-D18	
zener	zen-12v	-		12v	0	MG3-A11	
zener	zen-13v	-		13v	0	MG3-A12	

Figura 1.8 - risultati ricerca articoli a fine scorta

2. TECNOLOGIE UTILIZZATE

2.1 Scelta del sistema DBMS

Poiché per il progetto era necessario l'uso di un DBMS, sono state analizzate le caratteristiche dei sistemi database di uso più comune, alla ricerca di una soluzione adatta alle esigenze. Oggetto della ricerca sono stati Access e SQL Server di Microsoft, e MySQL di MySQL AB.

Microsoft Access fa parte del pacchetto Office, ed è un applicativo che consente di creare database in modo semplice ed intuitivo, seguendo procedure guidate dal programma. Access però presenta dei limiti nelle prestazioni "multiutente", soprattutto quando la mole di dati da gestire e il numero di connessioni contemporanee in scrittura salgono. Esiste una vera e propria diatriba in alcuni forum nel Web su quali siano le soglie di affidabilità di Access. Non ci sono certezze in assoluto, anche perchè i risultati dipendono dalla stabilità del progetto, in ogni modo è difficile pensare ad un accesso contemporaneo di 20 operatori in scrittura su un recordset senza che la connessione cada. Da questo punto di vista Access non regge il confronto con un vero e proprio DBMS come SQL server o MySQL.

Microsoft SQL Server è un DBMS molto completo, supporta il set di funzioni stabilite dallo standard SQL-99, e vanta prestazioni di tutto rispetto anche quando opera con ingenti quantità di dati e connessione di numerosi utenti. Questa soluzione è sicuramente adatta all'utilizzo in un progetto come quello trattato in questa sede. Il costo, però, non è trascurabile, e varia secondo la versione scelta (personal, standard o enterprise), richiedendo però sempre un certo esborso economico: una licenza server standard ha un costo di circa 900€, ma i prezzi variano considerevolmente secondo parametri quali il numero di client da installare, il numero di CPU del server, e il numero di licenze acquistate. Il costo riportato è indicativo, è riferito ad una licenza server comprensiva di dieci licenze client, ed è stato ottenuto dopo aver contattato un rivenditore, in quanto Microsoft non permette la consultazione on-line del listino.

MySQL è un DBMS di più recente introduzione, nasce da un progetto Open Source ed è utilizzato anche per gestire database molto complessi. L'insieme delle funzioni supportate è conforme alle specifiche SQL-92, ma sono state introdotte recentemente anche funzioni appartenenti a SQL-99. Si tratta dunque di un sistema molto meno completo rispetto a SQL Server, e indubbiamente è molto più complesso da configurare e programmare: non sono infatti forniti strumenti di lavoro integrati in ambiente grafico. L'indubbio vantaggio di MySQL rispetto ai concorrenti è il sistema di licenze: sotto certe condizioni l'uso è gratuito, e le licenze a pagamento sono comunque vendute a prezzi molto abbordabili. SQL è inoltre famoso per la velocità di esecuzione ottenuta anche su macchine server non molto potenti.

Le informazioni sul funzionamento e le prestazioni di MS SQL Server e MySQL sono trattate in modo esauriente all'interno della tesi *"Valutazione delle tecnologie XML, Web Service per l'interoperabilità tra DBMS relazionali",* di Yuri Debbi, dove si trova un'esauriente analisi delle prestazioni dei due sistemi.

Secondo quanto richiesto dall'azienda, a livello funzionale tutti i tre sistemi studiati sarebbero stati adatti per l'impiego nel progetto. Sono state quindi valutate altre caratteristiche.
Una differenza fondamentale tra le tre soluzioni è rappresentata dal costo: per un'applicazione come quella presa in esame l'uso di MySQL è gratuito, mentre le restanti soluzioni sono a pagamento. MySQL, inoltre, rappresenta un DBMS più completo ed affidabile rispetto ad Access, ed è paragonabile in questo senso a soluzioni professionali come SQL Server di Microsoft.

Il vantaggio risultante dall'impiego di MS Access sarebbe stato l'utilizzo più semplice: l'interfaccia di lavoro di questo sistema è, infatti, molto immediata, e l'interazione con pagine web dinamiche è gestita in modo automatico. Trattandosi però di un'esperienza di stage universitario, l'obiettivo era quello di familiarizzare con l'uso di sistemi professionali, e non si è ritenuto giusto cercare di perseguire la strada più semplice.

Se la scelta fosse ricaduta su SQL Server, infine, si avrebbe avuto un sistema simile a MySQL, con in aggiunta funzionalità più complesse, superflue al nostro utilizzo, e il non trascurabile svantaggio di un costo d'acquisto elevato.

La scelta di MySQL è apparsa quindi la più adatta allo scopo prefissato.

2.2 Introduzione a MySQL

Storia di MySQL

Nel 1996 un'azienda scandinava, la TcX, aveva bisogno di un database relazionale che potesse gestire grandi quantità di dati, ma che fosse allo stesso tempo molto veloce e sicuro e, come se non bastasse, che potesse essere usato con poche risorse hardware. In quel periodo, trovare un prodotto che avesse queste caratteristiche pareva impossibile. Allora l'azienda decise di creare il proprio DBMS, partendo dalla base di un altro database: il mSql. Crearono un prodotto veramente veloce e semplice, più veloce anche dei più costosi e "nobili" Oracle e SQL Server, ma privo di un'interfaccia veramente potente per la gestione manuale dei dati, come quelle disponibili per i database appena menzionati o per Access. MySQL è distribuito con un programma dal quale si può gestire il database, ma con un'interfaccia completamente testuale e che non ha nulla a che vedere con quelle dei principali concorrenti commerciali.

Nel tempo sono nati da gruppi di programmatori autonomi numerosi strumenti che implementano un'interfaccia grafica di facile utilizzo. Alcuni di questi software sono commerciali, alcuni altri sono gratuiti, come "DB Tool Manager", utilizzato per lo sviluppo del software oggetto di questa tesi.

Quelle descritte sopra non sono le uniche limitazioni di MySQL. Il sistema non supporta:

- right join
- chiavi esterne (supportate dalla prossima versione 5.0, in fase di beta testing)
- integrità referenziale.

Inoltre mancano altre caratteristiche che riguardano l'interazione con il linguaggio ASP. Funzionalità che non sono state integrate nel database per tenere alte le prestazioni in termine di velocità, possono in ogni caso essere aggirate da un bravo programmatore.

Un'importante funzione che non era disponibile fino alla versione 4.1 erano le query annidate (o sub-query). Fa parte della filosofia di MySQL AB di non introdurre nuove funzionalità fino a quando non si è ottenuta la massima efficienza, stabilità e velocità delle funzioni già presenti.

2.3 Caratteristiche di MySQL

1.Velocità

Il sistema MySQL nasce, per volontà degli sviluppatori, per essere quanto più "leggero" e veloce possibile. L'obiettivo è stato certamente raggiunto, ma il prezzo da pagare è la limitazione ad un insieme di funzioni ridotto rispetto ai prodotti concorrenti.

Gli sviluppatori dichiarano di aver implementato tutte le funzioni di cui ha bisogno la maggior parte degli utenti. Mancano delle funzioni, ma sono quelle che solo pochi utilizzatori sono interessati ad usare.

Le nuove funzioni sono sempre introdotte a ritmo di una o due per versione del programma, e sono sviluppate, ottimizzate e provate per mesi prima dell'effettiva introduzione, in modo da poter garantire la massima stabilità e la massima velocità di funzionamento.

Allo stato attuale delle cose, è disponibile la versione 4.1, che, rispetto alla precedente versione ha introdotto:

- Sub-query
- Tabelle derivate
- Set di caratteri internazionale UTF-8
- L'uso di una memoria "key cache" che aumenta la velocità d'accesso alle tabelle
- Un nuovo protocollo di trasmissione binario più veloce del precedente

La versione 5 del prodotto, che come detto è imminente, introduce altre novità, sempre all'insegna della velocità e della maggior completezza funzionale:

- Stored procedure
- Cursori
- Foreign key
- Funzioni OLAP (online analytical processing) per velocizzare le query su database complessi

[le caratteristiche riportate delle due versioni 4.1 e 5.0 sono prese dal documento "MySQL Metz" reperibile sul sito www.mysql.com]

2. Costi

Altro obiettivo di MySQL è quello di vendere un prodotto competitivo dal punto di vista economico: i costi, infatti, sono molto contenuti rispetto a quelli della concorrenza.
La licenza d'uso del prodotto dice:

" [...] free use for those who never copy, modify or distribute. As long as you never distribute the MySQL software in any way, you are free to use it for powering your application, irrespective of whether your application is under GPL license or not."

" […] uso gratuito per chi non copia, modificano o distribuiscono (l'applicazione). Finché non ridistribuite MySQL in alcun modo, siete liberi di usarlo per far funzionare la vostra applicazione, sia che questa sia o meno sotto licenza GPL.".

[fonte: "MySQL Open Source License" reperibile all'indirizzo: http://www.mysql.com/company/legal/licensing/opensource-license.html]

Questo ha reso possibile l'utilizzo gratuito del DBMS per l'applicazione oggetto della tesi.
Per coloro che utilizzano MySQL per creare applicazioni commerciali, invece, è previsto il pagamento della licenza; questa è proposta ad un prezzo più vantaggioso rispetto ai prodotti concorrenti, dai 250 o 500€ secondo la versione scelta, "classic" o "pro".

3. Portabilità

MySQL è completamente scritto e sviluppato tramite linguaggio C e C++ e verificato su vari compilatori in molteplici piattaforme. Questo sistema risulta attualmente essere l'unico DBMS disponibile per una così vasta scelta di piattaforme: allo stato attuale esistono versioni per Windows, MacOS, Solaris, FreeBSD, Novell NetWare, Linux (nelle sue innumerevoli incarnazioni) ed altri sistemi ancora.
Lo sviluppo in C/C++ e la libera distribuzione del codice sorgente ne rendono possibile la conversione per qualsiasi sistema esistente.

4. Open Source

Un'ultima caratteristica, che rende unico il sistema MySQL, è il sistema di sviluppo Open Source. Il codice che compone il programma è liberamente distribuito, ogni utente, purché esperto, può leggere e modificare il programma a proprio piacimento. Le modifiche possono essere inviate ai programmatori di MySQL AB, i quali valutano le modifiche e possono decidere di inserirle o meno in una successiva edizione del programma.
Grazie a questo lo sviluppo dell'applicazione procede in modo rapido, consentendo una costante evoluzione e una rapida correzione degli eventuali errori di programmazione introdotti lungo il percorso.
Questo sistema consente anche di mantenere un team ufficiale di programmatori interni molto ristretto, poco oneroso per la società, e in grado di prendere decisioni in modo rapido ed indipendente.

Si spiega così l'incredibile capacità di rendere disponibile un aggiornamento del prodotto circa una volta al mese: dal primo rilascio ufficiale della versione 4.1 si sono susseguiti ben sette aggiornamenti in soli otto mesi; tempi simili sono impensabili per le realtà delle grandi software-house.

2.4 Interfaccia utente

Per ottenere la massima semplicità di installazione ed uso del software, si è deciso di realizzare un'interfaccia web, sfruttando le potenzialità messe a disposizione dai linguaggi di scripting, che trasformano le comuni pagine HTML in veri e propri software applicativi.

A questo scopo esistono numerosi linguaggi ognuno con le proprie caratteristiche; i più noti ed utilizzati sono Perl, PHP, JSP e ASP.

Segue ora una breve panoramica su queste soluzioni, in particolare su ASP, il linguaggio che, per economia e semplicità, è stato scelto per il nostro progetto.

I linguaggi di scripting

Lo scopo dei linguaggi di scripting è quello di superare i limiti di staticità delle pagine HTML; senza farne uso non sarebbe possibile, per esempio, far sì che la pagina si modificasse secondo le preferenze di un utente, oppure fosse possibile raccogliere dati direttamente dal browser internet per memorizzarli su un database.

Esistono parecchi linguaggi di scripting, alcuni più simili a veri e propri linguaggi di programmazione, altri più facili.
Il perl, ad esempio, è il tipico caso di linguaggio complesso, ma molto efficace, utilizzato per la creazione di cgi (common gateway interface) a livello professionale e poco diffuso fra gli utenti medi, a causa della difficoltà di apprendimento e di gestione della sua sintassi; inoltre, essendo nato in ambiente Unix ha trovato non poche difficoltà ad affermarsi al di fuori di una pur vasta cerchia di provider e professionisti del settore che utilizzano questo sistema operativo.
Un'altra difficoltà notevole per l'utente medio è da sempre rappresentata dalla generale impossibilità di eseguire i cgi al di fuori della directory cgi-bin del web server del quale ci si serve, directory il cui accesso è limitato al web master.

Un altro linguaggio che si è molto diffuso è il PHP, che come ASP è un linguaggio di scripting, cioè si può inserire nelle pagine html ed ha delle notevoli potenzialità.
PHP, come Perl, nasce nel mondo Unix/Linux ma a differenza dell'ASP è un linguaggio più complesso.
Lo scopo degli attuali linguaggi è quello di rendere dinamiche le pagine web, mantenendo al contempo una semplicità di programmazione che consenta a tutti di intervenire senza prima dovere leggere voluminosi manuali.

JSP, creato da Sun Microsystem, è un linguaggio che ricorda molto da vicino ASP, sia nella sintassi che nel funzionamento. La principale differenza rispetto ASP è la possibilità di funzionare su piattaforme di qualsiasi tipo: JSP mantiene per la stessa politica adottata da Sun per il Java.

Active server pages (ASP)

ASP (acronimo di Active Server Pages) si distingue fra tutti i linguaggi di scripting per la rapidità e flessibilità di utilizzo che lo caratterizzano, che però sono controbilanciate da uno svantaggio non trascurabile; l'utilizzo di questo linguaggio è confinato ai server Microsoft come ad esempio a IIS, e non funziona quindi con tutti gli altri server che popolano il web (anche se è possibile usare server unix tramite software di emulazione recentemente introdotti).
La sempre maggiore diffusione dei server Windows contribuisce però a rendere meno limitante questo ostacolo e, tutto sommato, non è difficile vedere diversi provider abbandonare il mondo Unix per le nuove possibilità offerte dai sistemi server Microsoft.
Grazie all'utilizzo delle pagine asp, l'utente può quindi creare dei documenti che possono fornire informazioni, rispondendo in modo diverso alle differenti richieste dei navigatori.
Vediamo quali sono, in breve, i vantaggi nell'utilizzo di questo linguaggio di scripting:

1) Le pagine asp sono completamente integrate con i file html.
2) Sono facili da creare e non necessitano di compilazione.
3) Sono orientate agli oggetti e usano componenti server ActiveX.

Visti i vantaggi, e viste anche le limitazioni cui abbiamo accennato in precedenza, riassumiamo le tecnologie conivolte nello sviluppo e funzionamento delle active server pages:

1) Un sistema operativo Microsoft, come Windows 2000.
2) Protocollo tcp/ip
3) Un web server che supporti Active Server, come IIS (Internet Information Service)
4) In via facoltativa, un driver odbc (Open DataBase Connectivity) e un server database.

Esaminando più da vicino l'"anatomia" di questo genere di pagine si può constatare che esse sono costituite da tre differenti parti:

1) Testo
2) Marcatori html
3) Comandi script

In un documento con estensione .asp è consentito utilizzare variabili, cicli, istruzioni di controllo, e altri costrutti tipici dei linguaggi di programmazione, grazie alla possibilità di richiamare la sintassi un linguaggio di scripting, come ad esempio il vbscript e il javascript, ma anche perl e rexx.
La scelta del linguaggio dipende in primo luogo dalle necessità del programmatore e dal tipo di esecuzione che si vuole avere: se si vogliono eseguire gli script dal lato server è preferibile utilizzare il vbscript, mentre se ci si vuole affidare alla potenza degli "scripting engine" (motore che interpreta i comandi dei linguaggi di scripting e li

esegue) dei singoli navigatori è sicuramente meglio utilizzare il javascript, semplice ed efficace.

Accesso al database

Un aspetto fondamentale della programmazione di un'applicazione web-database è quello della comunicazione fra le pagine web e il sistema database. Fondamentalmente i metodi per effettuare questo collegamento sono due:

- Mediante la creazione di una stringa di collegamento, da inserire in ogni pagina web che ha accesso al database
- Tramite l'uso di un driver ODBC (Open DataBase Connectivity).

Utilizzando il primo metodo, si ha un vantaggio: ci si collega direttamente al driver del database installato sul server, senza la necessità di installare alcun software aggiuntivo. La stringa di connessione però è complessa, e nel caso si dovesse modificare il database installato sul server (per esempio per migrare verso un sistema diverso), sarebbe necessario modificare ogni singola pagina web, per aggiornare la stringa di connessione, che cambia nel driver di collegamento, negli username e password.

Il secondo metodo, invece, prevede l'installazione sul server di un driver aggiuntivo di tipo DBC (DataBase Connectivity). Tramite questo tipo di driver si crea una comunicazione standard tra il database e la pagina web. Il driver raccoglie il sistema di collegamento e i dati necessari alla connessione (compresi username e password); in questo modo, un'eventuale modifica futura al database non necessita di alcuna modifica alle pagine web.
Open Database Connectivity (ODBC) è una API (Application Program Interface) standard per la connessione ai DBMS. Questa API è indipendente dai linguaggi di programmazione dai sistemi di database e dal sistema operativo. ODBC permette ai programmi che lo usano di inviare ai database stringhe SQL senza che ci sia bisogno di conoscerne le API proprietarie. Genera automaticamente richieste che il sistema di database utilizzato sia in grado di capire.

Il nostro "Magazzino Laboratorio" utilizza questa soluzione, realizzata mediante un driver ODBC.
Dopo aver installato il driver ODBC (nel nostro caso myODBC, distribuito secondo licenza Open Source), è necessario configurare un DSN, ossia un "nome del database di origine"; tramite questo nome si indica il nostro database ogni volta che si effettua un collegamento ad esso.
Il codice ASP per il collegamento al database è il seguente:

```
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.ConnectionString = "DSN=dbmagazzino"
Conn.Open
```

```
set objRs = server.createobject("ADODB.Recordset")
```

Nella prima riga viene chiamato il metodo CreateObject, tramite cui si genera un'istanza della connessione al database. Nella seconda riga si indica, tramite il DSN, il nome del database al quale si sta effettuando il collegamento. Tramite il comando "conn.open" si apre la connessione al database. Nell'ultima riga si crea un oggetto "recordset" per la lettura e scrittura dei dati nel database.

Questa procedura viene ripetuta ogni volta che si vuole effettuare un collegamento al sistema. Una volta terminato l'accesso, è opportuno scollegarsi dal server, in modo di liberare la memoria occupata dalla sessione.

```
objRs.Close
set objRs = Nothing
Conn.Close
set conn = Nothing
```

Le prime due righe sono destinate alla chiusura dell'oggetto "recordset"; le due righe successive si occupano invece della chiusura della connessione vera e priopria col database server.

3. DESCRIZIONE DELL'APPLICAZIONE

3.1 Progettazione concettuale

La prima fase da affrontare per progettare un database è la "progettazione concettuale". Il suo scopo è quello di rappresentare la realtà di interesse in termini di una descrizione formale e completa, indipendentemente dai criteri di rappresentazione utilizzati nei DBMS.

Il progetto di questa fase è chiamato "schema concettuale" e fa riferimento ad un "modello concettuale dei dati"

Il più diffuso modello concettuale dei dati è il modello "Entity-Relationship" o "Entità - Relazione": a partire da documenti e moduli informativi sul problema, viene costruito uno schema ER, rappresentato da un diagramma, che descrive a livello concettuale la Base Dati.

Questa rappresentazione viene tradotta in uno schema relazionale (livello logico) costituito da una collezione di tabelle. Infine i dati sono descritti da un punto di vista fisico (tipo e dimensione dei campi), e vengono definite le strutture ausiliarie (indici) per l'accesso efficiente ai dati.

Nelle pagine seguenti è riportato lo schema E/R generale del database e quello degli utenti, accompagnati da una breve spiegazione.

Schema E/R generale

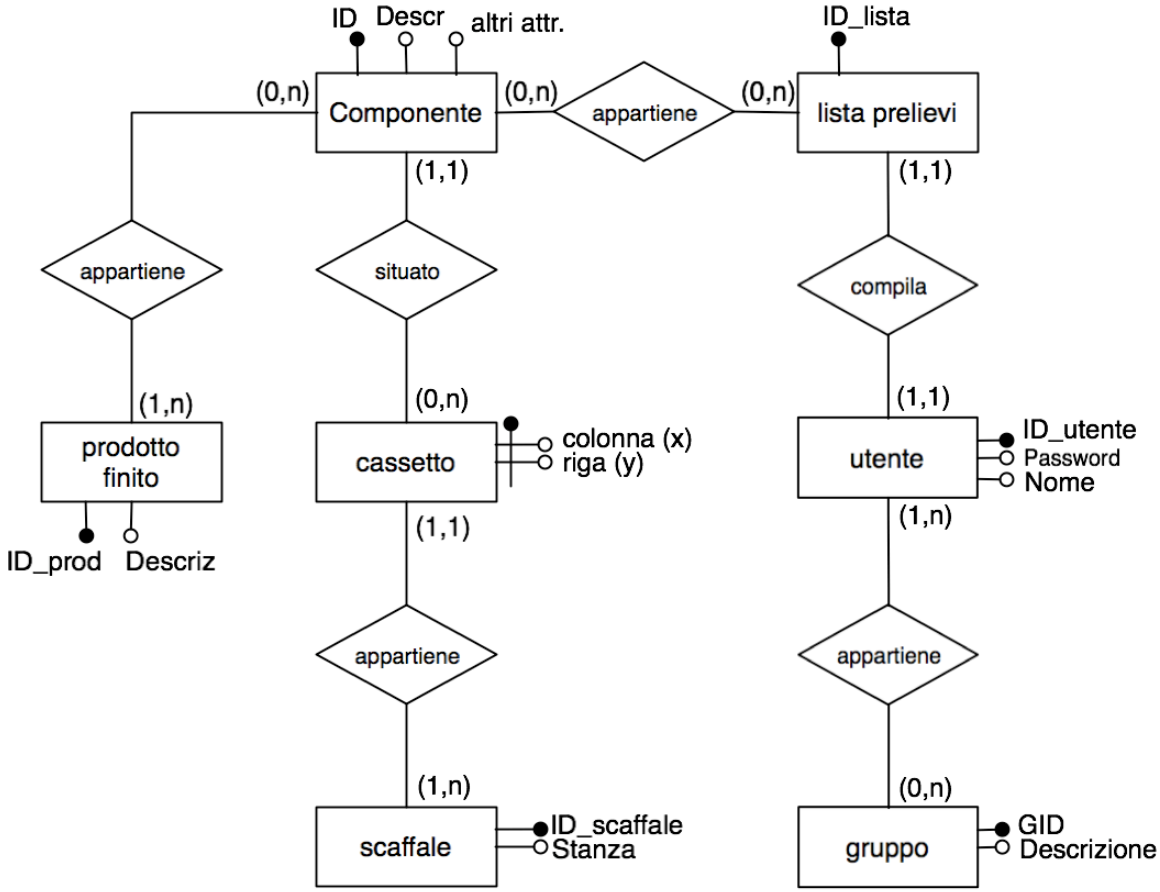


Figura 3.1 - Schema E/R del database creato

L'entità "componente" (che rappresenta il generico elemento contenuto in magazzino) viene identificato da un codice ID, che è primary key e quindi univoco. Gli altri attributi di "componente" sono il nome e una descrizione; altri attributi descrivono uno o più "valori" del componente (per esempio la capacità o la dimensione".

I componenti appartengono a una delle cinque categorie identificate ("tipo"), che possono essere "elettronica", "elettrica", "acustica", "meccanica" e "elettronica attiva".

Per facilitare la ricerca, si è stabilito di indicare anche l'appartenenza di un certo componente ad un prodotto finito. Ogni prodotto finito, dunque, è identificato dall'ID_prod (PK) e da una descrizione.

Ogni componente è collocato in un cassetto, identificato in modo univoco all'interno di una cassettiera mediante una numerazione dei cassetti per righe e colonne (colonna (x) e riga (y) costituiscono una Primary Key). Ogni cassettiera è identificata da un ID_scaffale (PK) e da una stanza in cui è collocata.

E/R: schema relazionale

Componente (Codice, Nome, Formato, Tipo, Valore, Collocazione)

Scaffale (ID_scaffale, Stanza)

Cassetto (Colonna (X), Riga (Y))
FK: ID_scaffale REFERENCES: Scaffale
FK: ID REFERENCES: Componente

Prodotto finito (ID_prod, Descr)

Gruppo (GID, Descrizione)

Utente (ID_utente, Password, Nome)
FK: GID REFERENCES: Gruppo

Lista prelievi (ID_Lista)
FK: ID REFERENCES: Componente

Schema ER: dettaglio della gestione utenti

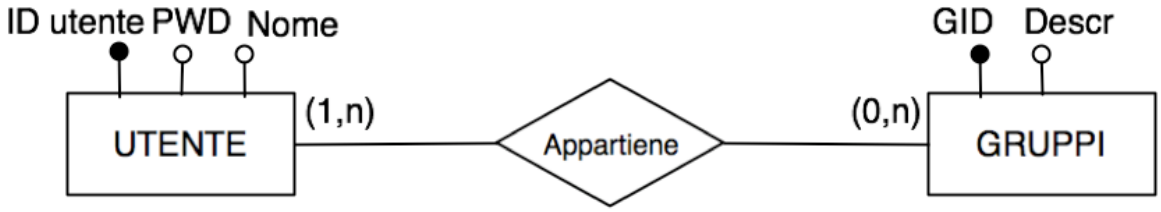


Figura 3.2- dettaglio dello schema E/R: utenti

Ogni utente è identificato da un ID_utente, che è univoco (Primary Key); gli altri attributi assegnati sono la password, nome e cognome completi. Gli utenti appartengono ad un gruppo, che può essere il gruppo degli utenti semplici o il gruppo degli amministratori. I gruppi sono contraddistinti da Identificativo Gruppo (GID) e da una descrizione.

La tabella "componenti" nel dettaglio

Poiché l'entità "componente" è caratterizzata da numerosi attributi, vediamo nel dettaglio come è stata creata la tabella relativa ai componenti.

Nome: varchar con una lunghezza massima di 30 caratteri alfanumerici; il campo non può essere lasciato nullo.

Codice: varchar con lunghezza 30 caratteri; è primary key.

Formato: varchar di massimo 30 caratteri;

Tipo: varchar di 15 caratteri; indica la categoria alla quale appartiene il prodotto.

Valore: varchar di 30 caratteri; può essere lasciato vuoto, per catalogare prodotti che non si caratterizzano secondo un valore.

Descrizione: varchar di massimo 100 caratteri; permette di fornire una descrizione articolata del prodotto inserito. Può essere lasciato vuoto.

Il comando SQL che permette di creare la tabella componenti è:

```
CREATE TABLE COMPONENTI
 (Nome varchar(30),
 Codice varchar(30) PRIMARY KEY,
 Formato varchar(30) NOT NULL,
 Tipo varchar(15) NOT NULL,
 Valore varchar(30),
 Descrizione varchar(100),
 CollocazioneX varchar(4) REFERENCES Cassetto(colonna),
 CollocazioneY varchar(4) REFERENCES Cassetto(riga));
```

3.2 Sicurezza e gestione delle sessioni

ASP: gestione delle sessioni

Accade spesso che, per il corretto funzionamento dell'applicazione, occorra mantenere vivo un dato per un tempo prefissato o fino a quando l'utente non si scollega dal programma. Esistono due metodi fondamentali per risolvere questo problema. Il primo è quello di passare di pagina in pagina una serie di valori tramite link e poi prelevarli nella pagina di destinazione. Questa soluzione, però, è da ritenersi scomoda e "poco sicura" in certe situazioni. La soluzione più logica allora è quella di inserire il valore da conservare all'interno di una variabile. L'unico inconveniente delle variabili è la loro visibilità. Infatti, è possibile vedere il loro valore solo all'interno della pagina di elaborazione. Esiste però un tipo particolare di variabile definita con l'oggetto Session. Come si intuisce dal nome stesso, la variabile definita come session vale per la sessione di lavoro attiva. Infatti ogni utente connesso, può leggere e variare solamente le proprie variabili di sessione. Si riporta la loro sintassi di definizione:

```
<%
' Attivazione di una variabile di sessione
   session("nome_variabile") = valore_da_assegnare
%>
```

É evidente come l'uso di una variabile di sessione sia abbastanza simile a quello di una variabile normale. L'unica differenza a livello di stesura del codice è quella della dichiarazione: il nome della variabile va racchiuso fra parentesi, virgolette e preceduta dalla parola "session"

```
   session(" ... ")
```

Per ogni variabile di sessione appena creata viene generato dal server un numero univoco di identificazione. Questo numero è progressivo e serve appunto al Server per riconoscere il valore della sessione ogni volta è utilizzata. La sintassi per visualizzare il numero Id di una sessione è:

```
<% response.write session.sessionId %>
```

Le variabili di sessione hanno un tempo di scadenza, oltre il quale vengono cancellate; il loro periodo di sopravvivenza dall'istante dell'attivazione è pari a 20 minuti. E' possibile variare questo tempo con la seguente sintassi:

```
<%
'Impostiamo la vita di sessione a n minuti
session.timeout = n %>
```

Tramite il codice appena illustrato, si dichiara che tutte le variabili di sessione dichiarate avranno vita n minuti. Allo scadere del tempo prefissato, la variabile sarà nulla e quindi non più esistente.

Nell'applicazione "Magazzino Laboratorio" le variabili di sessione sono state impostate per avere una vita di 40 minuti: questo consente di lasciare il programma in attesa anche per diverso tempo senza perdere i dati memorizzati.

È possibile procedere alla cancellazione o all'abbandono delle variabili prima che queste abbiano esaurito il loro tempo di vita. Col termine cancellazione, si intende lo svuotamento e quindi l'annullamento della variabile di sessione indicata. La sintassi per svolgere questo compito è la seguente:

```
<%
'Cancellazione di una variabile di sessione desiderata
   session("nome_variabile") = Null
%>
```

Assegnando ad una variabile di sessione il valore Null, questa variabile non esisterà più nel corso della sessione di navigazione dell'utente.

Con l'abbandono, a differenza della cancellazione, si cancellano tutte quante le variabili di sessione attive nell'istante in cui il comando viene incontrato. Per svolgere questa operazione bisogna utilizzare la seguente sintassi:

```
<%
' Abbandono di tutte le variabili di sessione
   session.abandon
%>
```

Gestione degli accessi al sistema

Nella pagina di login sono richieste all'utente ID e Password. Una query SQL cerca nel database se la corrispondenza esiste.

SELECT * FROM utenti WHERE nome=userID AND pwd=password

In seguito si è affrontato il problema di impedire l'accesso al sistema a coloro che non hanno effettuato il login: un estraneo potrebbe accedere direttamente alle singole pagine immettendo nella barra degli indirizzi del browser l'indirizzo diretto di una pagina. Per impedire questo, è stato inserito in ogni pagina il codice:

```
<%
if session("login") then

[… codice ASP e HTML della pagina…]

else
response.redirect "index.asp"
end if
%>
```

Il cui compito è quello di controllare che l'accesso sia stato effettuato correttamente, e in caso contrario provvede a mandare l'utente alla pagina di login.

3.3 Funzione di inserimento prodotti

Per inserire un articolo è necessario richiamare l'apposita funzione dal menu principale. Appare un form di inserimento, con un campo per ogni caratteristica del prodotto.

Una volta completata l'immissione dei campi, la pressione del tasto "inserisci" richiama la funzione contenuta nella pagina "inserisci_db.asp", che per prima cosa richiama i dati inviati dall'utente tramite la pagina precedente:

```
nome = request.form("nome")
 codice = request.form("codice")
 formato = request.form("formato")
 tipologia = request.form("tipologia")
 valore = request.form("valore")
 quantita = request.form("quantita")
 collocazione = request.form("collocazione")
```

Subito dopo viene calcolata la data attuale, in modo da poterla indicare nella tabella.

```
 data=day(now)& "-" & month(Now)&"-" & year(now)
```

In seguito avviene l'inserimento:

```
set objRs = Conn.Execute("INSERT into elettronica(nome, codice,
    formato, tipologia, valore, quantita, collocazione,
    ultima_mod)VALUES ('" &nome& "','" &codice& "','" &formato&
    "','" &tipologia& "','" &valore& "','" &quantita&"','"
    &collocazione& "','" &data&"')")
```

Dopo una prima fase di test, si è rilevato un malfunzionamento della funzione di ricerca dei prodotti: non venivano trovati i prodotti che erano stati inseriti lasciando vuoti i campi facoltativi (valore o descrizione). Questo è dovuto probabilmente al particolare funzionamento della sintassi "like" utilizzata nella funzione di ricerca. Si è risolto il problema inserendo nel campi vuoti il valore standard "-".

3.4 Funzione di ricerca

La funzione di ricerca che è stata implementata nell'applicazione prevede di specificare un numero variabile di parametri. Questa caratteristica è fondamentale, in quanto non sempre si conoscono tutti i dettagli dell'oggetto che si sta cercando; inoltre è di grande utilità poter cercare un insieme di oggetti che rispondono ad alcune determinate caratteristiche.

Per realizzare questo tipo di ricerca la query SQL creata è di questo tipo:

```
SELECT * from elettronica WHERE nome ='" & nome &"' AND valore
like '" &valore& "%' AND formato like '" &formato& "%'
```

Si è scelto di utilizzare la sintassi "like", per rendere più facile la ricerca; in questo modo non è necessario preoccuparsi della corretta digitazione dei parametri di ricerca: molto facilmente vengono trovati i risultati voluti anche quando l'immissione dell'utente è solo parziale. La controindicazione più ovvia, quella di ottenere risultati non coerenti con la ricerca, si è verificata raramente durante i test.

Al caricamento della pagina di ricerca, una query SQL si occupa di trovare tutti i nomi delle categorie di oggetti presenti nel database. In questo modo l'applicazione crea un menu di selezione a tendina. Questo aiuta l'utente nel velocizzare l'immissione, e impedisce che nel campo "nome" venga immesso un valore errato o non esistente.

Lo stesso sistema è stato adottato nella pagina di dei risultati della ricerca, per elencare il codice corrispondente ai prodotti trovati nella ricerca. Tramite questo menu si sceglie fra i prodotti trovati quale prelevare, aggiungere o modificare.

I risultati della ricerca vengono riportati all'utente in una tabella, che viene creata di dimensioni calcolate per contenere esattamente i risultati trovati. Il codice ASP utilizzato è:

```
<%
a=objRs.fields.Count
do while NOT objRs.EOF
 response.write "<tr>"
 for t=0 to a-1
  response.write "<td>" & objRs(t) & "</td>"
 next
 response.write "</tr>"
 objRs.MoveNext
loop
%>
```

3.5 Gestione dei prelievi di componenti

Funzionamento concettuale

Una parte fondamentale del programma “Magazzino Laboratorio” è la gestione del prelievo dei prodotti dal magazzino. Dopo aver compiuto una ricerca per parametri, il software propone all’utente un insieme di azioni che possono essere svolte sul prodotto selezionato; fra le azioni possibili c’è il prelievo di una quantità di prodotto dal magazzino.

Quando l’utente seleziona la quantità di elementi da prelevare, il programma ne verifica la disponibilità in magazzino, la confronta con la quantità richiesta dall’utente e, nel caso la richiesta eccedesse la quantità in scorta, genera un messaggio di errore.

Vediamo il codice ASP:

Inizialmente si interroga il database per sapere il numero di elementi in magazzino:

```
set objRs = Conn.Execute("SELECT quantita FROM elettronica
WHERE Codice='" &codice& "'")
```

In seguito si confronta tale quantità con quella richiesta dall’utente; se la quantità è insufficiente, viene richiamata una pagina di errore, e nessun prelievo è eseguito:

```
qta=objRs("quantita")
 risultato=qta-numero
 if risultato<0 then
 response.redirect "art_esaurito.asp"
 end if
```

Nel caso la quantità in scorta sia sufficiente a soddisfare la richiesta, avviene il prelievo.

```
set objRs = Conn.Execute("UPDATE elettronica SET
Quantita=Quantita-'" & numero & "', ultima_mod='"&data&"' WHERE
Codice='" & codice & "'")
```

Nella query di aggiornamento è aggiornata anche la data di ultima modifica. Questa funzione, non esplicitamente richiesta dall’azienda, è utile per risalire alla data in cui è avvenuto l’ultimo prelievo o l’ultimo inserimento: si evitano così possibili errori di doppio utilizzo.

Lista prelievi

Al termine di una sessione è di grande utilità poter consultare e stampare l'elenco dei prodotti prelevati. Per realizzare questa funzione è stata introdotta un'apposita tabella nel database, chiamata "prelievi"; la tabella contiene i campi del nome prodotto, il relativo codice e la quantità prelevata. Ogni volta che un utente preleva un elemento, il programma aggiorna la tabella, che può essere visualizzata in ogni momento. L'applicazione provvede ad eliminare il contenuto della lista prelievi al momento del login dell'utente. Questo accorgimento è indispensabile per assicurarsi che all'inizio della sessione la tabella sia vuota, e non contenga vecchi dati (per esempio rimasti da una sessione non terminata correttamente).

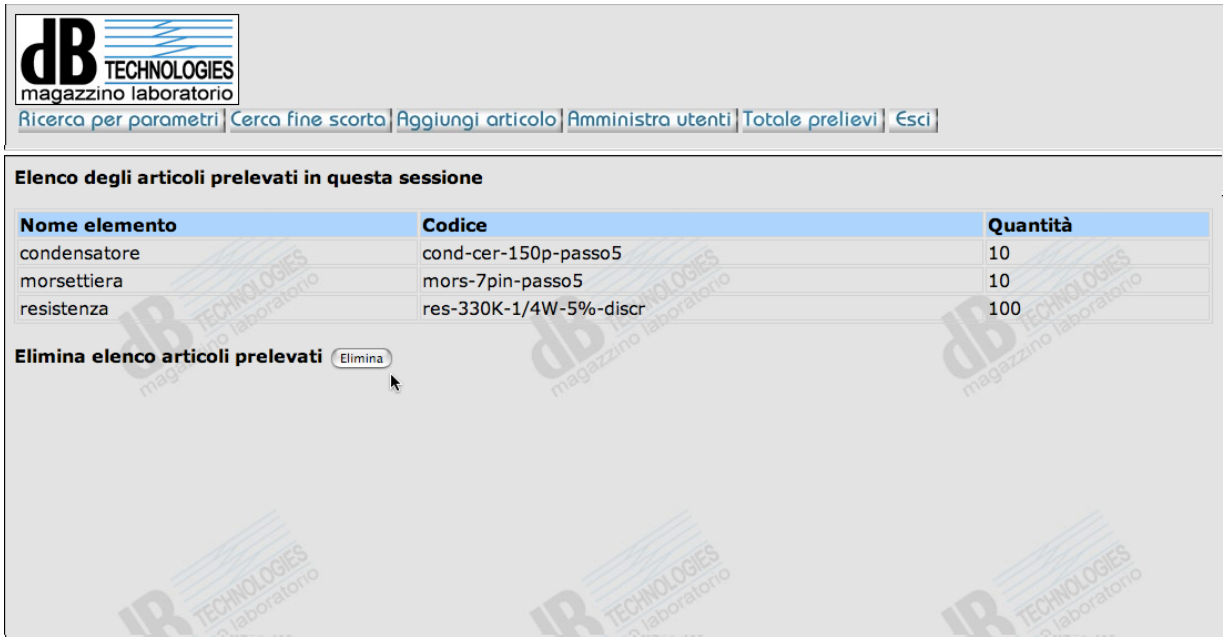


Figura 3.3 - lista dei prelievi eseguiti

3.6 Modifica di un elemento

Capita spesso di commettere errori nell'inserimento di un articolo, o di voler descrivere un determinato oggetto in modo più dettagliato ed esaustivo. È stata creata una funzione apposita per la modifica degli elementi, in modo da evitare la complicata operazione di eliminazione di un prodotto dal database e il successivo re-inserimento.

La funzione di modifica può essere chiamata dalla pagina dei risultati della ricerca, assieme alle funzioni "aggiungi" e "preleva". Quando si seleziona un prodotto da modificare, appare una pagina contenente i campi di inserimento delle caratteristiche dell'oggetto; una query al database richiama tutti i dati relativi al prodotto, che vengono inseriti come valori di default nella pagina.

Questo facilita notevolmente la procedura di modifica, in quanto non è necessario modificare ogni singolo campo, ma solamente quelli ritenuti sbagliati.

Si riporta il codice ASP corrispondente. Inizialmente si leggono le caratteristiche correnti dal database:

```
<%
set objRs = Conn.Execute("SELECT * from elettronica WHERE
codice ='" & codice &"'")
nome=objRs("nome")
codice=objRs("codice")
formato=objRs("formato")
tipologia=objRs("tipologia")
valore=objRs("valore")
quantita=objRs("quantita")
collocazione=objRs("collocazione")
%>
```

in seguito sono visualizzati i campi di immissione, il cui valore di default è il valore corrente, contenuto nel database

```
<%
<input type="text" name="nomeF" size="10" maxlength="30"
value="<%response.write(nome)%>">

input type="text" name="codiceF" size="10" maxlength="30"
value="<%response.write(codice)%>"></td>

[... e così via per i diversi campi di descrizione ]
%>
```

Alla conferma da parte dell'utente, si procede all'aggiornamento del database.

4. CONCLUSIONI

L'obiettivo del progetto è stato quello di creare un'applicazione che potesse facilitare la gestione di un piccolo magazzino. Si è creato un software semplice da usare, dotato delle funzioni di principale interesse, e adatto allo scopo prefissato.

Per lo sviluppo sono stati utilizzati software "Open Source" o "freeware", dimostrando così la possibilità di realizzare applicazioni anche corpose pur mantenendo una spesa contenuta. Gli svantaggi derivanti da questo tipo di soluzione consistono nella maggior difficoltà d'uso e nella minore completezza delle funzioni offerte: svantaggi che, vista la relativa semplicità dell'applicazione creata, sono stati facilmente superati.

L'applicazione è stata realizzata durante il periodo di stage, sempre a diretto contatto con i futuri utenti: numerose funzioni sono state aggiunte, rimosse o modificate durante lo sviluppo, sempre seguendo le indicazioni che man mano sono state date dall'azienda. L'attività di stage è stata quindi molto formativa, in quanto ho potuto verificare l'importanza di un'attenta pianificazione, da svolgersi nelle fasi iniziali del progetto, ancora prima di iniziare la stesura del programma vero e proprio.

L'esperienza provata mi ha permesso di conoscere in modo abbastanza approfondito il database MySQL, sistema relativamente recente, ma che sta raccogliendo sempre più consensi, grazie alla continua evoluzione e al prezzo contenuto: un numero sempre crescente di *internet provider* sta adottando questa soluzione per offrire spazio web per siti dinamici a basso costo.
Ho inoltre potuto migliorare notevolmente la conoscenza della programmazione web-oriented e ho approfondito la conoscenza delle Active Server Pages (ASP).
La stessa stesura di questa tesi mi ha consentito di impiegare e approfondire le conoscenze apprese durante il corso di diploma.

Problemi riscontrati
Durante la fase di sviluppo e la successiva fase di test, sono stati riscontrati problemi, dovuti spesso ad incompatibilità fra il sistema ASP e MySQL; questi malfunzionamenti sono stati risolti o aggirati, ma può essere interessante vederli nel dettaglio:

- Nelle pagine ASP che eseguono accessi al database non è stato possibile eseguire comandi JavaScript: lo script veniva ignorato e proseguiva l'esecuzione della pagina. Questa situazione ha dato qualche difficoltà in quelle funzioni in cui si voleva richiedere una conferma da parte dell'utente. Il problema è stato risolto inserendo una pagina di intermezzo, dove è stato inserito lo script voluto;

- Fra le caratteristiche che descrivono i componenti ci sono dei campi la cui immissione non è obbligatoria: se per un certo prodotto uno di questi campi veniva lasciato vuoto, tale prodotto non veniva riportato fra i risultati della ricerca nel database. Si è risolto impostando come valore di default il simbolo " - " per i campi opzionali.

- Gli utenti hanno la frequente abitudine di chiudere il browser per terminare il lavoro, invece di servirsi dell'apposito tasto "esci" Non è possibile impedire all'utente di chiudere il browser mentre la sessione è aperta: questo ha reso complicata la gestione delle sessioni e della tabella dei prelievi, in quanto il browser non ha modo di chiudere le variabili aperte nel momento in cui il programma viene chiuso. Per questo motivo le tabelle dei prelievi e le variabili di sessione vengono inizializzate prima del login, in modo da evitare malfunzonamenti.

Possibili sviluppi futuri

Al termine del progetto è stato preparato un manuale d'uso dell'applicazione, il cui scopo è guidare l'utente passo per passo all'installazione e alla configurazione del sistema, e all'uso del programma. L'ultimo capitolo di tale guida è dedicato alla struttura del codice dell'applicazione: l'intento è quello di fornire il massimo supporto possibile per eventuali sviluppi futuri del programma. A tale scopo ogni pagina di codice è stata ampiamente commentata, e ogni funzione è stata spiegata nel dettaglio.
Questi accorgimenti sono stati presi in quanto l'applicazione manca ancora di accorgimenti che possono risultare utili nell'utilizzo quotidiano. Per esempio si potrebbe rendere più flessibile la gestione dei prodotti a fine scorta: allo stato attuale la funzione esegue una ricerca sui prodotti che sono presenti in un numero minore di quello indicato; una soluzione più utile è quella di poter indicare un valore limite differente per ogni tipo di prodotto in magazzino. Un possibile sviluppo futuro potrebbe essere quello di creare una piccola applicazione in grado di installare e configurare in modo automatico il server MySQL, e che si occupi di impostare il backup delle tabelle del database ad orari stabiliti: attualmente questo lavoro è svolto da un programma esterno, inserito in fase di installazione, e configurato manualmente dall'utente.

BIBLIOGRAFIA

DOCUMENTAZIONE ONLINE

"**Guida completa alle ASP**" di Andrea Carratta, pubblicata sul sito www.html.it all'indirizzo http://freeasp.html.it/guide/lezioni.asp?idguida=4 . La guida, articolata in 46 lezioni, è stata di prezioso aiuto per apprendere le conoscenze fondamentali delle ASP.

"**MySQL Reference Manual**", documento di oltre 1500 pagine, che cura ogni aspetto di MySQL: dalle caratteristiche fino all'installazione sui principali sistemi. Ogni informazione riguardo MySQL nella presente tesi è documentata dal succitato manuale.

TESTI SCRITTI

Progetto di Basi di Dati Relazionali: Lezioni ed Esercizi
Autori: Beneventano, Bergamaschi, Vincini –
Editore: Pitagora Editrice - Bologna

MySQL Tutorial
Autori: Laura Thomson, Luke Welling
Editore: Pearson Education Italia

ASP
Autore: Elijah Lovejoy
Editore: Tecniche Nuove

TESI CONSULTATE
Durante l'attività di stage e di stesura del presente elaborato, sono state di prezioso aiuto alcune tesi di laurea, tutte discusse nell'Ateneo di Modena e Reggio Emilia:

- *"Analisi e prototipazione di una interfaccia utente che consenta la realizzazione di semplici pagine web sulla base di template"* di Lorenzo Ballasini, AA 2003-2004
- *"Sviluppo di applicazioni Web-DB a due e tre livelli: valutazione comparativa"* di Filippo Battilani, AA 2001-2002
- *"Valutazione delle tecnologie XML, Web Service per l'interoperabilità tra DBMS relazionali",* di Yuri Debbi, AA 2002-2003

GLOSSARIO

API: API è l'acronimo di Application Program Interface, indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per un determinato compito. È un metodo per ottenere un'astrazione, di solito tra l'hardware e il programmatore, o tra software a basso ed alto livello. Le API permettono di evitare ai programmatori di scrivere tutte le funzioni dal nulla.

DBMS: Abbreviazione di DataBase Management System, è un programma informatico (o, più frequentemente, un insieme di programmi) progettato per gestire un database, ovvero un insieme di dati strutturati.

Freeware: Il termine freeware indica un software che viene distribuito in modo gratuito. Tipicamente il freeware è distribuito senza codice sorgente e contiene una licenza che ne permette la redistribuzione; può però avere delle restrizioni. A volte la licenza garantisce la libera copia del programma, ma non la vendita.

ODBC: Acronimo di Open Database Connectivity, ODBC è una API standard per la connessione ai DBMS. Questa API è indipendente dai linguaggi di programmazione dai sistemi di database e dal sistema operativo. ODBC si basa sulle specifiche di Call Level Interface (CLI) di SQL, X/Open e ISO/IEC. E' stata creata dall'SQL Access Group e la sua prima release risale al settembre 1992.

Open Source: In informatica, open source (termine inglese che significa sorgente aperto) indica un software rilasciato con un tipo di licenza per la quale il codice sorgente è lasciato alla disponibilità di eventuali sviluppatori, in modo che con la collaborazione (in genere libera e spontanea) il prodotto finale possa raggiungere una complessità maggiore di quanto potrebbe ottenere un singolo gruppo di programmazione. L'open source ha ovviamente tratto grande beneficio da internet, e spesso si lega a principi ideali di gratuità.

SQL: acronimo di Structured Query Language. Linguaggio creato per l'accesso a informazioni memorizzate nei database. L'SQL nasce nel 1974 nei laboratori dell'IBM. Nasce come strumento per lavorare con database che seguano il modello relazionale. L'ANSI lo adottò come standard fin dal 1986. Nel 1987 la ISO fece lo stesso. Questa prima versione standard è denominata SQL/86. Negli anni successivi sono state realizzate altre versioni, SQL/89, SQL/92 e SQL/2003.