UNIVERSITÀ DEGLI STUDI DI PALERMO

FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Tesina di Ingegneria del Software

Sistema informatico per la Gestione di una Libreria Universitaria

Docenti:

Prof. Massimo Cossentino
Prof. Umberto Lo Faso

Allievi Ingegneri:

Daniele Ribaudo
Leonardo Papuzza
Valerio Finazzo

Anno Accademico 2002/2003

Indice Generale

Software Project Management Plan

Revision History:

Versione R0.1, creazione: 27 Agosto 2002.
Versione R1.0, modifica: 29 Novembre 2002.

Preface:

Questo documento definisce il processo tecnico e la pianificazione della produzione del sistema; esso è rivolto a: sviluppatori, analisti e clienti.

Target Audience:

Sviluppatori, Analisti, Clienti

Project Members:

Valerio Finazzo, Leonardo Papuzza, Daniele Ribaudo

Indice

1. Introduzione

BOOKBASE rappresenta un software per la gestione di una libreria universitaria che si interfaccia con gli impiegati della libreria per supportarli nello svolgimento delle attività tipiche, che sono:

- vendita dei libri;
- gestione del magazzino;
- acquisizione dei libri.

Il software prevede l'utilizzo di 3 postazioni, una per l'addetto alle vendite, una per la gestione del magazzino e una per la gestione degli ordini; ogni postazione presenterà la porzione di software relativa ai compiti che dovrà svolgere.

1.1. Overview del progetto

Questo documento è rivolto alla descrizione degli aspetti manageriali e tecnici ed è finalizzato alla pianificazione ed alla programmazione.
Le fasi del progetto e le attività cardine sono mostrate nella tabella seguente; ogni fase produce un relativo documento che deve essere approvato dal responsabile del progetto e dai clienti.

Data	Fase del Progetto	Attività cardine
27 Agosto 2002	Studi preliminari	
29 Agosto 2002	Analisi	
13 Settembre 2002		Presentazione specifiche al committente
13 Settembre 2002	Progettazione	
26 Settembre 2002	Implementazione	
21 Ottobre 2002		Presentazione del software versione 0.9
21 Ottobre 2002	Testing	
11 Novembre 2002		Revisione finale
13 Novembre 2002	Rilascio del software versione 1.0	
03 Dicembre 2002		Consegna del prodotto

Tabella 1 - Schedulazione del Progetto

Per ulteriori approfondimenti sugli argomenti si rimanda ai paragrafi successivi.

1.2. Documenti per il cliente

Il seguente progetto produrrà un software per la gestione di una libreria universitaria che dovrà supportare tutte le funzionalità specificate nel documento di analisi dei requisiti (RAD) discusso in seguito.
La seguente documentazione del progetto sarà prodotta e fornita al cliente:

- **Software Project Management Plan**: definisce il processo tecnico e la pianificazione della produzione del sistema (questo documento).

- **Requirements Analysis Document**: descrive le funzionalità del sistema da realizzare; esso è formato da quattro tipi di modelli: *use case model*, *object model*, *functional model* e *dynamic model*. Tale documento è creato interagendo con gli esperti del dominio d'applicazione ed è approvato dal cliente.

- **System Design Document**: descrive gli obiettivi, l'architettura, la decomposizione ad alto livello del sistema, le piattaforme hardware/software, le condizioni di confine. Il documento fornisce la base per l'implementazione del sistema e per la stesura del progetto esecutivo.

- **Object Design Document**: descrive il sistema in termini di oggetti, loro metodi e attributi. Questo documento viene redatto durante la fase di implementazione del sistema.

- **Manuale Utente**: descrive il funzionamento del software.

- **Codice sorgente** del sistema.

- **Pacchetto di installazione del software**: sarà fornito un CD contenente il programma eseguibile che installa e configura in maniera automatica il sistema nella macchina dell'utente.

1.3. Evoluzione del Software Project Management Plan

La pianificazione delle attività e l'allocazione delle risorse sono gestite mediante il pacchetto Microsoft Project XP. Eventuali cambiamenti relativi alla pianificazione e alle sue risorse saranno notificati e questo documento sarà aggiornato.

1.4. Materiale di riferimento

I riferimenti utilizzati per la realizzazione del software BOOKBASE sono riportati di seguito:

- Bernd Bruegge, Allen H. Dutoit, "*Object-Oriented Software Engineering*", Prentice Hall, (2000)
- Librerie MSDN (Pacchetto Visual Studio)
- Francesco Balena, "*Visual Basic 6*", Mondadori informatica
- Guida in linea dell'applicazione Rational Rose 2002 Enterprise Edition
- Rick Dobson, *"Programmare Access"*, Mondadori informatica
- Specifiche UML ver. 1.3

1.5. Acronimi e Definizioni

CASE - Computer Aided Software Engineering
GUI - Graphical User Interface
ODD - Object Design Document
RAD - Requirements Analysis Document
ROSE - Visual modeling tool
SDD - System Design Document
SPMP - Software Project Management Plan
UML - Unified Modeling Language
SQL - Structured Query Language
ADO - ActiveX Data Object

2. Organizzazione del Progetto

2.1. Processo di Riferimento

Il progetto è iniziato il 27 Agosto 2002 ed è terminato il 03 Dicembre 2002. Le principali attività cardine sono state:

- presentazione delle specifiche al committente: 13 Settembre 2002;
- presentazione del software dimostrativo v0.9: 21 Ottobre 2002;
- revisione finale del software: 11 Novembre 2002;
- consegna del prodotto: 03 Dicembre 2002.

Il progetto adotta una metodologia Object-Oriented basata sul ciclo di vita del software a cascata e utilizza UML come linguaggio di riferimento. Il processo di sviluppo è organizzato in diverse attività. Le attività e i milestones sono illustrate nelle sezioni seguenti.

2.1.1. Studi preliminari

Gli studi preliminari comprendono l'acquisizione dei concetti necessari per lo sviluppo del software; le principali tematiche trattate riguardano:

UML	→	per la stesura di tutta la documentazione;
DataBase	→	per lo sviluppo e implementazione degli archivi utilizzati dal software;
Rational Rose	→	strumento CASE utilizzato per lo sviluppo del progetto;
Visual Basic	→	per l'implementazione dei moduli e dell'interfaccia grafica.

2.1.2. Analisi dei requisiti

L'attività di analisi dei requisiti esamina il problema posto in termini di consistenza, completezza e fattibilità. Questa attività è caratterizzata dalla continua interazione con i clienti, interazione che consente di determinare il modello dei requisiti: un insieme di modelli del sistema proposto. Il modello dei requisiti si articola in quattro parti fondamentali:

- *use case model*,
- *object model,*
- *functional model,*
- *dynamic model.*

La specifica di questi modelli consente di descrivere tutte le funzionalità del sistema, le sue prestazioni, le interfacce con gli altri elementi, i vincoli.

2.1.3. Pianificazione del progetto

La fase di pianificazione del progetto include la descrizione degli obiettivi, delle attività, delle dipendenze e delle risorse impiegate nel progetto. Il risultato di questa fase è il *software project management plan* (SPMP, questo documento). Completata l'attività di progettazione, la fase di pianificazione fornisce anche il contratto per il cliente.

2.1.4. Presentazione specifiche

L'attività comprende la presentazione delle specifiche di progetto al cliente per eventuali revisioni e l'approvazione delle funzionalità del sistema. Tale presentazione è stata svolta in data 13 Settembre 2002.

2.1.5. Progettazione

Lo scopo dell'attività di progettazione è concepire un'architettura di sistema che tracci il modello ottenuto in fase di analisi con l'ambiente scelto. La parte fondamentale di questa fase è la progettazione di sottosistemi, che rappresenta la suddivisione del sistema in esame coerentemente alle piattaforme scelte per l'utilizzo. In questa attività inoltre vengono perfezionati gli use cases prodotti in fase di analisi e descritte le interazioni fra gli oggetti in ogni specifico use case.

2.1.6. Implementazione

L'attività ha come obiettivo la codifica (nel linguaggio scelto) dei singoli moduli e oggetti descritti nell'*object design document* (ODD).

2.1.7. Presentazione del software dimostrativo

Questa attività riguarda la presentazione del software dimostrativo e la verifica di tutte le funzionalità del sistema al cliente; in questa fase si concordano con il cliente eventuali revisioni

del software. La prova di funzionamento è avvenuta in data 21 Ottobre 2002.

2.1.8. Testing

L'attività comprende la fase di Alpha-Test che avviene all'interno dell'azienda e la fase di Beta-Test che avviene all'esterno dell'azienda; le eventuali revisioni e correzioni apportate al software comporteranno il rilascio di versioni successive.

2.1.9. Revisione finale

Questa attività comprende la verifica finale del software ed il rilascio della versione 1.0.

2.1.10. Rilascio del prodotto

L'attività consente la creazione del pacchetto completo che comprende la stesura del manuale utente e la creazione del software di installazione; il pacchetto sarà fornito in CD.

2.1.11. Consegna del prodotto

L'attività riguarda la presentazione del prodotto finale e la dimostrazione delle funzionalità del software al cliente.

2.2. Risorse umane impiegate

Al progetto sono assegnate diverse entità, con profili professionali e capacità differenti; le varie entità costituiscono un singolo team. Le professionalità inserite si suppongono essere sufficienti per la realizzazione dell'intero progetto.
Le figure coinvolte sono:

- **Responsabile del progetto**: è colui che coordina e supervisiona tutto il progetto organizzando le risorse umane, incontri con il cliente e incontri dell'intero team per revisioni e discussioni. Ad esso è assegnata la responsabilità del successo o del fallimento del progetto.

- **Analista**: si occupa di fornire un'adeguata analisi del problema avvalendosi di esperti del settore e consulenti esterni; il suo scopo è tradurre in specifiche tecniche le richieste del cliente.

- **Ingegnere del Software**: è colui che ha maggiore esperienza nella pianificazione e nella progettazione; ad egli viene assegnato il compito di pianificare l'intero progetto, nel rispetto dei vincoli imposti dal cliente (economici e temporali).

- **Programmatore**: ha il compito di implementare i componenti del sistema.

- **Progettista GUI**: ha il compito di sviluppare le interfacce grafiche del sistema.

- **Tester**: ha il compito di testare il sistema e di comunicare ai programmatori eventuali errori riscontrati.

Per l'allocazione delle risorse umane nelle varie attività del progetto si vedano la tabella e i diagrammi riportati di seguito (diagramma di Gantt e diagramma PERT).

ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Studi preliminari**	**2,67 days**	**Tue 27/08/02**	**Thu 29/08/02**		
2	UML	0 days	Tue 27/08/02	Tue 27/08/02		Ingegnere del Software;Responsabile del Progetto;Analista
3	DataBase	1,33 days	Tue 27/08/02	Wed 28/08/02	2	Responsabile del Progetto;Analista;Ingegnere del Software[25%]
4	Rational Rose 2002	2,67 days	Tue 27/08/02	Thu 29/08/02	2	Ingegnere del Software[75%]
5	Visual Basic	0,5 days	Tue 27/08/02	Tue 27/08/02	2	Programmatori;Progettista Gui
6	**Analisi**	**11,17 days**	**Thu 29/08/02**	**Fri 13/09/02**	**1**	
7	Analisi del problema	3,33 days	Thu 29/08/02	Tue 03/09/02		Analista;Responsabile del Progetto[10%];Ingegnere del Software[50%]
8	Raccolta requisiti	3,33 days	Wed 04/09/02	Mon 09/09/02	7	Responsabile del Progetto;Cliente[75%]
9	Stesura documento RAD	0,5 days	Mon 09/09/02	Mon 09/09/02	8	Responsabile del Progetto;Ingegnere del Software;Analista;Addetto alla Documentazioni
10	Revisione con il committente	2 days	Mon 09/09/02	Wed 11/09/02	9	Cliente;Responsabile del Progetto[50%]
11	Modifiche documento RAD	4 days	Mon 09/09/02	Fri 13/09/02	9	Responsabile del Progetto[50%];Ingegnere del Software;Analista
12	Pianificazione del progetto	6 days	Thu 29/08/02	Fri 06/09/02		Ingegnere del Software[50%];Responsabile del Progetto[90%]
13	Presentazione specifiche al committente	0 days	Fri 13/09/02	Fri 13/09/02	6	Responsabile del Progetto;Cliente;Consulente Legale
14	**Progettazione**	**9 days**	**Fri 13/09/02**	**Thu 26/09/02**	**13**	
15	Progetto del DataBase	6 days	Fri 13/09/02	Mon 23/09/02		Analista
16	Decomposizione del sistema	5 days	Fri 13/09/02	Fri 20/09/02		Ingegnere del Software
17	Stesura documento SDD	0,67 days	Fri 20/09/02	Mon 23/09/02	16	Ingegnere del Software;Responsabile del Progetto;Addetto alla Documentazioni
18	Progettazione GUI	2 days	Mon 23/09/02	Wed 25/09/02	17	Progettista Gui
19	Stesura documento ODD	1,33 days	Wed 25/09/02	Thu 26/09/02	18	Ingegnere del Software;Responsabile del Progetto;Addetto alla Documentazioni
20	**Implementazione**	**15 days**	**Thu 26/09/02**	**Thu 17/10/02**	**14**	
21	implementazione GUI	5 days	Thu 26/09/02	Thu 03/10/02		Progettista Gui;Programmatori
22	implementazione Database	6 days	Thu 26/09/02	Fri 04/10/02		Analista;Programmatori
23	implementazione componenti del software	15 days	Thu 26/09/02	Thu 17/10/02		Programmatori
24	Presentazione versione 0.9 del software	0 days	Mon 21/10/02	Mon 21/10/02	20	Responsabile del Progetto;Cliente;Consulente Legale
25	**Testing**	**15 days**	**Mon 21/10/02**	**Fri 08/11/02**	**24**	
26	Alfa Test	4 days	Mon 21/10/02	Thu 24/10/02		Programmatori;Tester
27	Beta Test	11 days	Fri 25/10/02	Fri 08/11/02	26	Tester
28	Revisione finale	2 days	Mon 11/11/02	Tue 12/11/02	25	Responsabile del Progetto;Ingegnere del Software;Analista
29	**Rilascio del prodotto**	**15 days**	**Wed 13/11/02**	**Tue 03/12/02**	**28**	

Figura 1 - Tabella di pianificazione

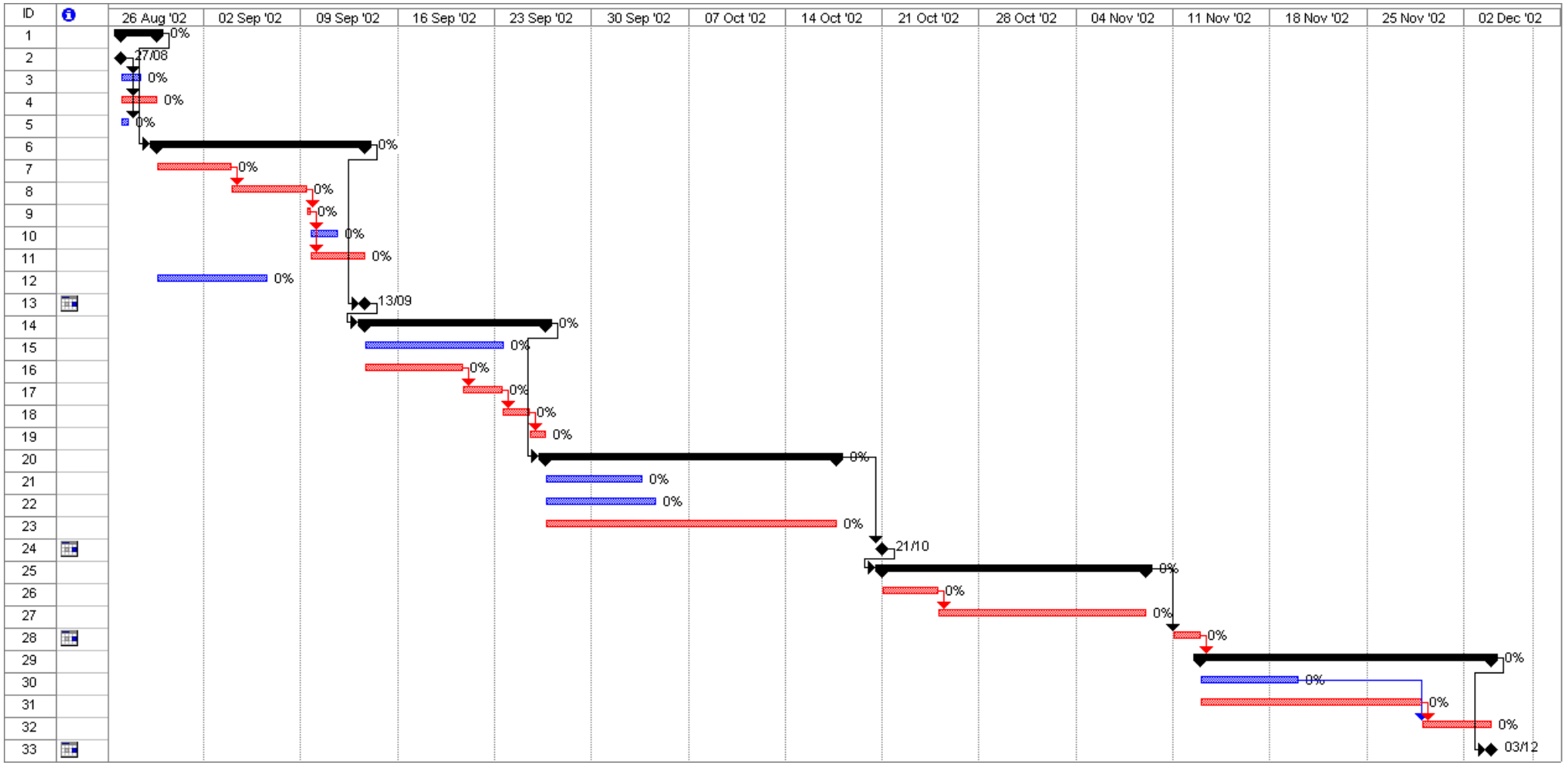
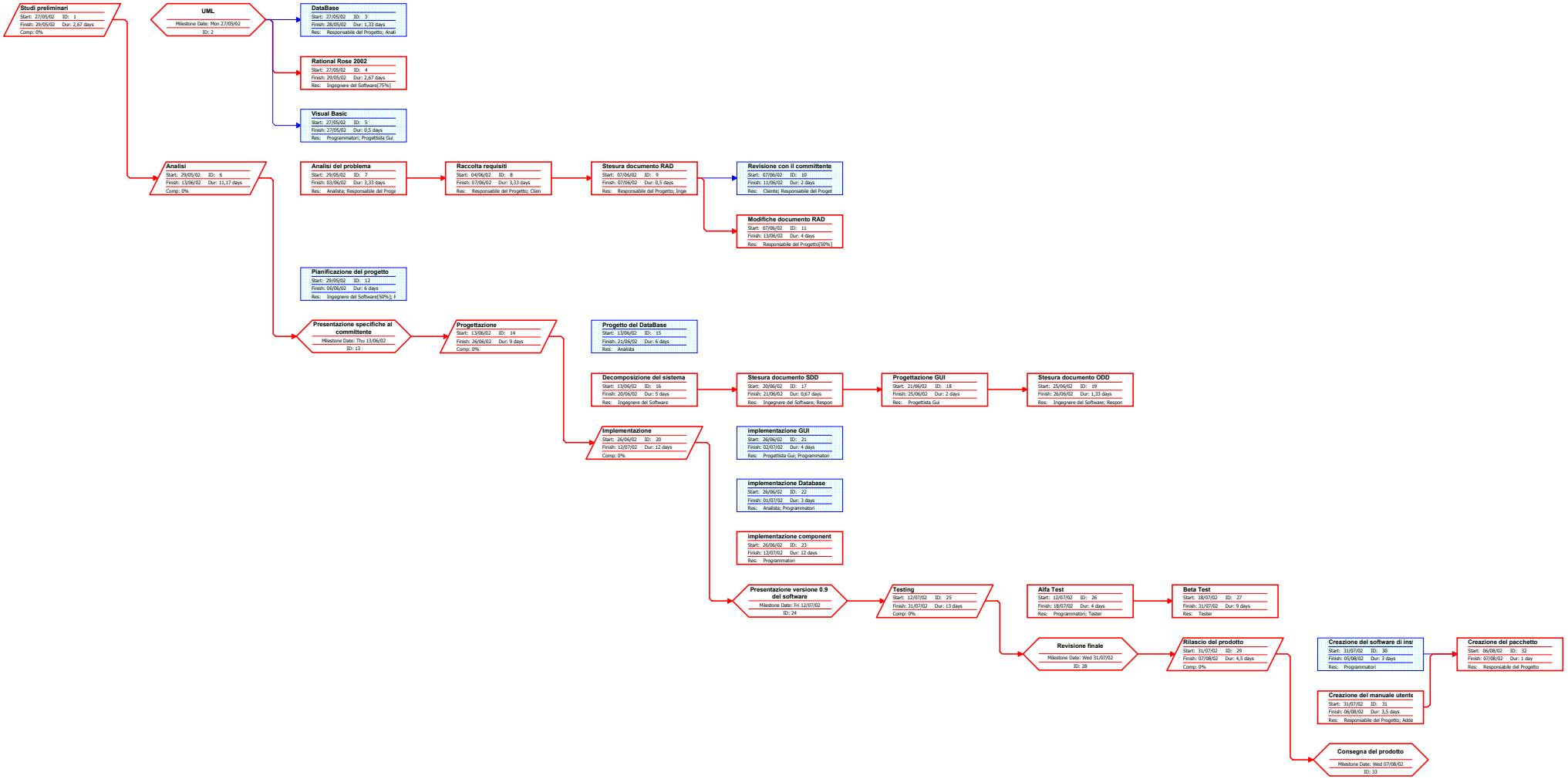


Figura 2 - Diagramma di Gantt


Studi preliminari
Start: 27/05/02 ID: 1
Finish: 29/05/02 Dur: 2,67 days
Comp: 0%
UML
Milestone Date: Mon 27/05/02
ID: 2
DataBase
Start: 27/05/02 ID: 3
Finish: 28/05/02 Dur: 1,33 days
Res: Responsabile del Progetto; Anali
Rational Rose 2002
Start: 27/05/02 ID: 4
Finish: 29/05/02 Dur: 2,67 days
Res: Ingegnere del Software[75%]
Visual Basic
Start: 27/05/02 ID: 5
Finish: 27/05/02 Dur: 0,5 days
Res: Programmatori; Progettista Gui
Analisi
Start: 29/05/02 ID: 6
Finish: 13/06/02 Dur: 11,17 days
Comp: 0%
Analisi del problema
Start: 29/05/02 ID: 7
Finish: 01/06/02 Dur: 3,33 days
Res: Analista; Responsabile del Proge
Raccolta requisiti
Start: 04/06/02 ID: 8
Finish: 07/06/02 Dur: 3,33 days
Res: Responsabile del Progetto; Clien
Stesura documento RAD
Start: 07/06/02 ID: 9
Finish: 07/06/02 Dur: 0,5 days
Res: Responsabile del Progetto; Inge
Revisione con il committente
Start: 07/06/02 ID: 10
Finish: 11/06/02 Dur: 2 days
Res: Cliente; Responsabile del Proget
Modifiche documento RAD
Start: 07/06/02 ID: 11
Finish: 13/06/02 Dur: 4 days
Res: Responsabile del Progetto[50%]
Pianificazione del progetto
Start: 29/05/02 ID: 12
Finish: 06/06/02 Dur: 6 days
Res: Ingegnere del Software[50%]; I
Presentazione specifiche al committente
Milestone Date: Thu 13/06/02
ID: 13
Progettazione
Start: 13/06/02 ID: 14
Finish: 26/06/02 Dur: 9 days
Comp: 0%
Progetto del DataBase
Start: 13/06/02 ID: 15
Finish: 21/06/02 Dur: 6 days
Res: Analista
Decomposizione del sistema
Start: 13/06/02 ID: 16
Finish: 20/06/02 Dur: 5 days
Res: Ingegnere del Software
Stesura documento SDD
Start: 20/06/02 ID: 17
Finish: 21/06/02 Dur: 0,67 days
Res: Ingegnere del Software; Respon
Progettazione GUI
Start: 21/06/02 ID: 18
Finish: 25/06/02 Dur: 2 days
Res: Progettista Gui
Stesura documento ODD
Start: 25/06/02 ID: 19
Finish: 26/06/02 Dur: 1,33 days
Res: Ingegnere del Software; Respon
Implementazione
Start: 26/06/02 ID: 20
Finish: 12/07/02 Dur: 12 days
Comp: 0%
implementazione GUI
Start: 26/06/02 ID: 21
Finish: 02/07/02 Dur: 4 days
Res: Progettista Gui; Programmatori
implementazione Database
Start: 26/06/02 ID: 22
Finish: 01/07/02 Dur: 3 days
Res: Analista; Programmatori
implementazione component
Start: 26/06/02 ID: 23
Finish: 12/07/02 Dur: 12 days
Res: Programmatori
Presentazione versione 0.9 del software
Milestone Date: Fri 12/07/02
ID: 24
Testing
Start: 12/07/02 ID: 25
Finish: 31/07/02 Dur: 13 days
Comp: 0%
Alfa Test
Start: 12/07/02 ID: 26
Finish: 18/07/02 Dur: 4 days
Res: Programmatori; Tester
Beta Test
Start: 18/07/02 ID: 27
Finish: 31/07/02 Dur: 9 days
Res: Tester
Revisione finale
Milestone Date: Wed 31/07/02
ID: 28
Rilascio del prodotto
Start: 31/07/02 ID: 29
Finish: 07/08/02 Dur: 4,5 days
Comp: 0%
Creazione del software di ins
Start: 31/07/02 ID: 30
Finish: 05/08/02 Dur: 3 days
Res: Programmatori
Creazione del pacchetto
Start: 06/08/02 ID: 32
Finish: 07/08/02 Dur: 1 day
Res: Responsabile del Progetto
Creazione del manuale utente
Start: 31/07/02 ID: 31
Finish: 06/08/02 Dur: 3,5 days
Res: Responsabile del Progetto; Adde
Consegna del prodotto
Milestone Date: Wed 07/08/02
ID: 33

3. Tool di sviluppo

Il progetto è stato sviluppato con l'ausilio di strumenti software alcuni dei quali propriamente dedicati all'ingegneria del software. Si riporta di seguito l'elenco di tali strumenti:

- **Rational Rose 2002**: strumento CASE utilizzato nella progettazione dei modelli del sistema (RAD), nella scomposizione del sistema (SDD) e nella definizione degli oggetti del sistema (ODD).
- **Microsoft Project XP**: strumento utilizzato per la pianificazione del progetto.
- **Visual Basic 6.0**: strumento (e linguaggio) utilizzato per l'implementazione degli oggetti del sistema e dell'interfaccia grafica.
- **Adobe Acrobat 5.0**: strumento utilizzato per la stesura della documentazione del sistema.
- **Microsoft Visio 2002**: strumento utilizzato per la progettazione del DataBase del sistema.
- **Microsoft Access XP**: strumento utilizzato per l'implementazione del DataBase del sistema.

Requirements Analisys Document

Revision History:

Versione R0.1, creazione: 29 Agosto 2002.
Versione R0.2, modificato 10 Settembre 2002.
Versione R0.6, modificato 23 Settembre 2002.
Versione R0.8, modificato 12 Ottobre 2002.
Versione R0.9, modificato 21 Novembre 2002.
Versione R1.0, modificato 06 Dicembre 2002.

Preface:

Questo documento indica i requisiti e le funzionalità del progetto BOOKBASE; esso è rivolto agli sviluppatori e ai clienti del progetto.

Target Audience:

Sviluppatori, Cliente

Project Members:

Valerio Finazzo, Leonardo Papuzza, Daniele Ribaudo

Indice

1. Obiettivi General

Il presente progetto si propone la realizzazione di un sistema informatico in grado di rendere più agevole ed efficiente la gestione di una libreria universitaria.
Attività principe di una libreria universitaria è rifornire gli studenti di libri di testo adottati dai vari corsi nel più breve periodo possibile. In genere fra l'inizio di un corso universitario e la disponibilità dei libri da esso adottati intercorrono alcune settimane; in situazioni come queste gli studenti cercano di sopperire a tale mancanza acquistando i libri di testo da librerie virtuali su Internet (es.: Amazon.com), che presentato tempi di consegna tipicamente di quattro giorni lavorativi e prezzi vicini a quelli di una comune libreria. Il sistema proposto è stato progettato e sviluppato al fine di colmare questo 'gap' e di recuperare tale fetta di mercato, di rilievo per una libreria di medio-piccole dimensioni.

2. Sistema Corrente

Al momento attuale si suppone che la libreria universitaria in esame gestisca il commercio dei libri senza alcun supporto d'automazione; tale libreria è comunque in relazione di stretta collaborazione con l'università che rifornisce; qualora l'università non abbia già provveduto, sarà compito del responsabile della libreria richiedere i tabulati degli iscritti ai vari corsi universitari e dei testi adottati e consigliati.
Si stima che i tempi di consegna relativi alle forniture dei libri siano dipendenti dalla loro eventuale disponibilità presso i fornitori nella misura media di tre giorni lavorativi per i testi disponibili e di quindici giorni lavorativi per i testi non disponibili.
Si suppone infine che tutto il sottosistema di rilascio fatture/scontrini è gestito manualmente secondo una metodologia di lavoro classica.

3. Sistema Proposto

BOOKBASE è un pacchetto software per la gestione di una libreria universitaria che si interfaccia con i suoi impiegati per supportarli nello svolgimento delle attività tipiche della stessa, che sono:

- vendita di libri;
- prenotazione di libri;
- gestione del magazzino;
- acquisizione dei libri.

Scopo strategico del software è quello di snellire il carico di lavoro degli impiegati della libreria e di fornire ai suoi clienti un servizio più efficiente, proponendosi di coadiuvare gli impiegati nelle abituali attività solitamente gestite secondo una metodologia di lavoro classica.
Il software si propone di rendere il sistema gestionale della libreria quanto più possibile automatizzato e di semplice utilizzo, al fine di sostituire nel breve periodo il sistema attuale poco efficiente e privo di qualsiasi possibilità di pianificazione delle attività di acquisto.
Al fine di perseguire gli obiettivi esposti, il sistema deve:

- ♦ guidare gli utenti nelle varie operazioni;

- ♦ fornire le informazioni richieste ed effettuare le operazioni di prenotazione, di vendita, di gestione magazzino e ordini in maniera corretta;

- ♦ consentire al gestore un completo controllo sulle attività dell'esercizio, per assicurare un servizio sempre più efficiente ai clienti, ed un continuo abbattimento dei costi di gestione, condizioni peraltro garantite dalla possibilità di ottimizzazione delle operazioni relative all'esercizio stesso.

Si ritiene che il sistema possa essere utilizzato da un'utenza media, è quindi di fondamentale importanza predisporlo per un facile utilizzo da parte di persone tipicamente non abituate a convivere con programmi di gestione informatici; al contempo è probabile che gli eventuali fruitori abbiano già un'esperienza sufficiente a comprendere tutti i passi che si succedono nello svolgimento di operazioni di gestione di un esercizio informatizzato.
Per tali motivi il prodotto è stato modellato in maniera da essere ne eccessivamente semplice, ne eccessivamente complesso.

3.1. Overview

Il sistema informatico che si vuole realizzare intende automatizzare la gestione delle principali attività svolte dal personale di una generica

libreria; si è volutamente omesso il riferimento esplicito ad una libreria universitaria in quanto lo stesso sistema può essere adottato da una libreria che tratta libri di qualsiasi genere. Tuttavia alcune funzionalità del sistema sono state ottimizzate per essere utilizzate nell'ambito universitario come la gestione di libri associati a dei corsi universitari. Queste funzionalità verranno discusse con maggiore dettaglio nel seguito del documento.
Il sistema in esame si inserisce nel contesto del contatto diretto con l'utenza nell'ambito dello svolgimento delle operazioni di consultazione delle giacenze, di vendita o prenotazione di libri, di gestione del magazzino e di gestione degli ordini di una libreria:

- la ricerca di un libro e delle sue informazioni è l'attività che può essere considerata più comune; tale funzionalità consente di fornire un servizio più efficiente ai clienti e più comodo all'addetto vendite, il quale è svincolato dal riferirsi continuamente all'addetto al magazzino;

- per la vendita di libri si deve tener conto della possibilità del cliente di acquistare o prenotare, se questi non sono disponibili, uno o più articoli;

- la gestione del magazzino comprende operazioni estremamente semplici; in particolare saranno semi-automatiche nel caso di consegne di libri (il magazziniere avrà semplicemente una form da compilare) ed automatiche nel caso di vendita degli stessi (il sistema si occuperà dell'aggiornamento delle giacenze);

- la gestione degli ordini presuppone che essi siano effettuati unicamente dal responsabile della libreria; questi potrà comunicarli ai relativi fornitori via telefono o via fax; avrà inoltre la possibilità di accedere in qualsiasi momento alle informazioni relativi agli ordini non ancora evasi, per eventuali solleciti al fornitore;

- il supporto alle decisioni fornisce gli strumenti che consentono al responsabile di ottimizzare l'approvvigionamento dei singoli libri; la raccolta delle informazioni sarà effettuata in base al periodo dell'anno, che andrà eventualmente confrontato con le date di svolgimento del corso, in base alle giacenze di magazzino, alle eventuali prenotazioni ed alle vendite effettuate fino alla data di interesse.

3.2. Requisiti Funzionali

Il software prevede l'utilizzo di 3 postazioni, una per l'addetto alle vendite, una per la gestione del magazzino e una per la gestione degli

ordini; ogni postazione utilizza la porzione di software relativamente ai compiti che dovrà svolgere, in particolare:

- la postazione *Vendite*, usata probabilmente da un commesso, comprende tutte le funzionalità riguardanti la consultazione delle giacenze di magazzino, la registrazione delle vendite, la richiesta di prenotazioni di libri;

- la postazione *Magazzino*, usata probabilmente da un magazziniere, comprende tutte le funzionalità riguardanti la registrazione delle consegne di libri; si noti che un ordine si considera evaso quando tutti i libri in esso contenuti sono stati presi in consegna;

- la postazione *Gestione Ordini*, usata probabilmente dal responsabile del punto vendita, comprende tutte le funzionalità riguardanti la registrazione degli ordini, l'accesso agli ordini non evasi, ed il supporto alle decisioni.

Il sistema proposto deve rispondere inoltre ai seguenti requisiti minimi:

- tenere aggiornato un database dei corsi (nome, numero studenti, modulo di svolgimento) e dei libri in essi adottati (che possono essere "consigliati" o "richiesti");

- gestire i movimenti di magazzino per avere una visione sempre aggiornata delle giacenze;

- mantenere una storia dell'acquisto/vendita dei libri;

- fornire un supporto alle decisioni per ottimizzare l'approvvigionamento di libri (a partire dall'analisi del periodo dell'anno, numero di studenti e giacenze di magazzino).

3.3. Requisiti non Funzionali

3.3.1. Interfaccia Utente e Fattori Umani

Si ritiene che il sistema possa essere utilizzato da un'utenza media, è quindi di fondamentale importanza predisporlo per un facile uso da parte di persone tipicamente non abituate a convivere con programmi di gestione informatici.
Al contempo è probabile che gli eventuali fruitori abbiano già un'esperienza sufficiente a comprendere tutti i passi che si succedono nello svolgimento di operazioni di gestione di un esercizio informatizzato. Per tali motivi il prodotto è stato modellato in modo

da essere ne eccessivamente semplice, ne eccessivamente complesso.

3.3.2. Documentazione

La seguente documentazione del progetto sarà prodotta e fornita al cliente:

- **Software Project Management Plan**: definisce il processo tecnico e la pianificazione della produzione del sistema.
- **Requirements Analysis Document**: descrive le funzionalità del sistema da realizzare; esso è formato da quattro tipi di modelli: *use case model*, *object model*, *functional model* e *dynamic model*. Tale documento è creato interagendo con gli esperti del dominio d'applicazione ed è approvato dal cliente (questo documento).
- **System Design Document**: descrive gli obiettivi, l'architettura, la decomposizione ad alto livello del sistema, le piattaforme hardware/software, le condizioni di confine. Il documento fornisce la base per l'implementazione del sistema e per la stesura del progetto esecutivo.
- **Object Design Document**: descrive il sistema in termini di oggetti, loro metodi e attributi. Questo documento viene redatto durante la fase di implementazione del sistema.
- **Manuale Utente**: descrive il funzionamento del software.
- **Codice sorgente** del sistema.
- **Pacchetto di installazione del software**: sarà fornito un CD contenente il programma eseguibile che installa e configura in maniera automatica il sistema nella macchina dell'utente.

3.3.3. Considerazioni Hardware

I terminali necessari al sistema proposto non presentano particolari richieste hardware per assolvere alle funzionalità implementate nel software.
Di seguito si suggerisce, senza voler imporre determinati vincoli di scelta, dei modelli di componenti hardware ritenuti adatti alle necessità:

- N° 3 Terminali (Personal Computer Assemblati):
  - Processore: AMD o INTEL based (>1 Ghz);
  - Ram: 256 Mb;
  - HDD: 40 Gb per le postazioni vendite e magazzino, 80 Gb per la postazione ordini;
  - Scheda Video: Agp con almeno 32 Mb di memoria;
  - CD-RW 24x10x40;
  - Scheda Ethernet 100 MBit/s;
  - Modem/Fax V.90, 4 USB, 1 Parallela, 1 Seriale;
  - Monitor 15" LCD;
- N° 3 Stampanti Laser, una connessa al terminale di gestione, una connessa al terminale vendite e l'altra al terminale magazzino;
- N° 1 Hub Fast Ethernet

- Cavo UTP, plug-in RJ45 e terminatori sufficienti al cablaggio dell'intera rete;

3.3.4. Prestazioni Hardware

Non essendo state effettuate richieste esplicite di requisiti non funzionali da parte del committente, visto che il sistema informativo non presenta particolari esigenze né in termini di un eventuale funzionamento multipiattaforma, né in termini di eventuali richieste di capacità "real-time", non sono richieste particolari prestazioni della macchina dell'utente se non quelle necessarie per installare BOOKBASE. I requisiti consigliati per installare tale software sono quelli riportati in [*3.3.3 Considerazioni Hardware*].

3.3.5. Interfaccia del Sistema

Il software non richiede dispositivi di interfacciamento aggiuntivi particolari alla macchina dell'utente; Esso fa uso dei dispositivi I/O tradizionali di un Personal Computer: monitor, tastiera, mouse, stampante.

3.4. Vincoli (Pseudo Requisiti)

L'ambiente di sviluppo utilizzato nella implementazione di BOOKBASE è Visual Basic poiché è quello più indicato nelle applicazioni che non presentano grosse elaborazioni ed il cui compito consiste nell'interrogare e mostrare le informazioni estratte da una base di dati. Visual Basic possiede a tal proposito la possibilità di inserire e configurare facilmente oggetti ADO per la connessione da e verso un database ed è stato concepito per creare interfacce utente windows like in breve tempo.

Le eventuali librerie utilizzate nell'implementazione del software saranno fornite insieme al pacchetto di installazione che si occuperà di collocarle all'interno delle macchine dell'utente.

3.5. Modelli del Sistema

3.5.1. Scenari

Di seguito si riportano alcuni scenari tipici che si presentano nell'utilizzo di BOOKBASE:

Scenario name	**Ricerca_libro_di_un_corso**
Partecipating actor instances	Marco: Commesso Gigi: Cliente
Flow of events	1. Gigi richiede a Marco i libri adottati dal corso di Ingegneria del Software del prof. M. Cossentino; 2. Marco inserisce i dati relativi al corso e conferma; 3. Marco scorre tutti i libri adottati dal corso, richiesti e consigliati; 4. Marco fornisce tutte le informazioni a Gigi;

Scenario name	**Inserimento_libro_corso**
Partecipating actor instances	Marco: Commesso
Flow of events	1. Marco dispone delle informazioni relative ai corsi e ai libri adottati; 2. Marco apre la finestra di inserimento e inserisce i dati relativi ai corsi; 3. Marco inserisce i dati relativi ai libri; 4. Marco specifica la relazione tra libri e corsi; 5. Marco salva le informazioni e chiude la finestra di inserimento.

Scenario name	**Vendita_del_libro**
Partecipating actor instances	Marco: Commesso Gigi: Cliente
Flow of events	1. Gigi richiede a Marco il libro di riferimento del corso di Ingegneria del Software del prof. M. Cossentino; 2. Marco effettua la ricerca dalla sua postazione e BOOKBASE lo informa che il libro è disponibile; 3. Gigi vuole acquistare il libro; 4. Marco salva la vendita e rilascia il documento di vendita; 5. Gigi ritira il libro ed esce dalla libreria.

Scenario name	**Prenotazione_del_libro**
Partecipating actor instances	Marco: Commesso Gigi: Cliente
Flow of events	1. Gigi richiede a Marco il libro di riferimento del corso di Ingegneria del Software del prof. M. Cossentino; 2. Marco effettua la ricerca dalla sua postazione e BOOKBASE lo informa che il libro non è presente in magazzino; 3. Gigi vuole prenotarne una copia e rilascia il nominativo e un acconto; 4. Marco inserisce questi dati nell'apposita form e stampa la ricevuta di prenotazione, che conterrà anche tutte le informazioni sul libro richiesto; 5. Gigi conserva la ricevuta ed esce dalla libreria.

Scenario name	**Registrazione_vendita_post_prenotazione**
Partecipating actor instances	Marco: Commesso Gigi: Cliente
Flow of events	1. Gigi da a Marco la ricevuta di prenotazione; 2. Marco consulta la lista delle prenotazioni, ricercando la voce relativa alla prenotazione effettuata da Marco e il sistema lo informa della disponibilità del libro in questione; 3. Marco quindi richiede a Gigi il saldo effettua la registrazione della vendita e stampa la ricevuta di vendita; 4. Marco consegna a Gigi il libro e la ricevuta di vendita; 5. Gigi conserva la ricevuta ed esce dalla libreria.

Scenario name	**Carico_libri_ordinati**
Partecipating actor instances	Carlo: Magazziniere
Flow of events	1. Carlo apre il pacco appena consegnato dal corriere e ne controlla il contenuto confrontandola con il D.D.T. (Documento di Trasporto); 2. Avendo constatato che tutte le voci presenti nel documento coincidono con il contenuto del pacco, Carlo effettua dalla sua postazione la ricerca dell'ordine, basandosi sulla data di effettuazione dell'ordine e sul fornitore; 3. BOOKBASE fornisce una form contenente un riepilogo dei libri costituente l'ordine; 4. Carlo specifica i libri e le relative quantità che sono stati consegnati; 5. Carlo conferma e invia le informazioni al sistema.

Scenario name	**Ordina_libri_di_un_corso**
Partecipating actor instances	Lucio: Responsabile
Flow of events	1. Lucio sta dando un'occhiata ai moduli dei corsi che si attiveranno nei prossimi 15 giorni. BOOKBASE gli fornisce per ogni corso tutte le informazioni necessarie per scegliere se effettuare alcuni ordini (libri richiesti, libri consigliati, numero di studenti), basandosi ad esempio sul numero dei libri venduti nello stesso corso negli anni passati e dalle giacenze di magazzino; 2. Lucio decide di effettuare l'ordine di 30 libri del corso di Ingegneria del Software del prof. Cossentino che inizierà tra 10 giorni; 3. BOOKBASE fornisce a Lucio le informazioni relative al fornitore della casa editrice del libro succitato; 4. Completati l'immissione dei dati Lucio effettua l'ordine e invia le informazioni a BOOKBASE che provvederà ad immagazzinare l'ordine nel database relativo agli ordini pendenti.

Scenario name	**Cancellazione_ordine**
Partecipating actor instances	Lucio: Responsabile
Flow of events	1. Lucio sta dando un'occhiata alla lista degli ordini ancora non evasi e si accorge che un ordine relativo a due settimane fa non è stato completamente evaso; 2. BOOKBASE fornisce a Lucio le informazioni relative al fornitore dell'ordine in questione. 3. Lucio contatta telefonicamente il fornitore per un sollecito ma egli gli risponde che la consegna avrà ancora un ritardo di 1 mese, causa ristampa nuova edizione. 4. Lucio decide di annullare l'ordine, invia tali informazioni a BOOKBASE, eliminando tutte le voci superflue.

Scenario name	**Analisi_giacenze_di_magazzino**
Partecipating actor instances	Lucio: Responsabile
Flow of events	1. Lucio apre la form relative alle giacenze di magazzino; 2. BOOKBASE interroga il DB e fornisce a Lucio, per ogni articolo, le informazioni relative alle giacenze, con in evidenza le giacenze che sono al di sotto di una soglia prestabilita; 3. Lucio decide di effettuare un ordine di uno di questi articoli in esaurimento; 4. Egli quindi seleziona tale articolo e il sistema gli fornisce le informazioni immediate per il riordino.

Scenario name	**Analisi_history_acquisti/vendite**
Partecipating actor instances	Lucio: Responsabile
Flow of events	1. Lucio apre la form relative all'history acquisti/vendite; 2. BOOKBASE interroga il DB e visualizza la lista dei titoli in cui è presente la quantità venduta e quella acquistata in riferimento al periodo in questione; 3. Lucio nota che in tale periodo si è verificato un trend inaspettato di elevata vendita del libro di Ing. Del Software e decide di effettuare un riordino di tale articolo; 4. Egli seleziona tale articolo e il sistema gli fornisce le informazioni immediate per il riordino.

Scenario name	**Analisi_vendite**
Partecipating actor instances	Lucio: Responsabile
Flow of events	1. Lucio apre la form relative alle analisi vendite; 2. il sistema interroga il DB in base ai filtri impostati da Lucio (data a partire dalla quale effettuare l'analisi); 3. BOOKBASE riporta i risultati ottenuti su una lista contenenti per ogni titolo le informazioni sul libro, il numero di copie vendute e la percentuale $\frac{QV}{G}$ ovvero quantità venduta su giacenza di magazzino nel periodo di interesse. Tale informazione fornisce la porzione di libri in giacenza che è possibile vendere nell'ipotesi di riconferma del trend di vendita precedente.

Scenario name	**Registrazione_di_un_fornitore**
Partecipating actor instances	Lucio: Responsabile
Flow of events	1. Lucio deve inserire nel database i dati di un nuovo fornitore con cui vuole intraprendere rapporti commerciali; 2. Egli apre l'apposita finestra e specifica l'operazione di inserimento di un nuovo fornitore; 3. La form richiede di inserire tutti i dati del fornitore e le case editrici che distribuisce; 4. Lucio inserisce tutti i parametri necessari e salva le impostazioni nel sistema.

Scenario name	**Ricerca_e_modifica_di_un_fornitore**
Partecipating actor instances	Lucio: Responsabile
Flow of events	1. Lucio è stato avvisato del fatto che un fornitore ha modificato la lista di editori che distribuisce; 2. Egli apre l'apposita finestra per effettuare la ricerca del fornitore in questione; 3. Lucio inserisce i dati per la ricerca del fornitore e riceve una lista di fornitori; 4. Egli seleziona il fornitore e visualizza i dati in una finestra apposita; 5. Lucio effettua le opportune modifiche sulla lista degli editori distribuiti dal fornitore; 6. Infine conclude l'operazione salvando le modifiche nel sistema.

Scenario name	**Ordine_libri_prenotati**
Partecipating actor instances	Lucio: Responsabile
Flow of events	1. Lucio deve effettuare un ordine ed ha la necessità di sapere quali libri richiedere ed in quale quantità; 2. Egli decide di occuparsi dei libri richiesti su prenotazione dai clienti; 3. L'esito della ricerca fornisce un insieme di libri e le quantità prenotate; 4. Lucio effettua l'ordine dei libri fissando le quantità da ordinare per ogni libro e specificando i dati del fornitore; 5. Lucia salva i dati dell'ordine nel sistema.

3.5.2. Use Case Models

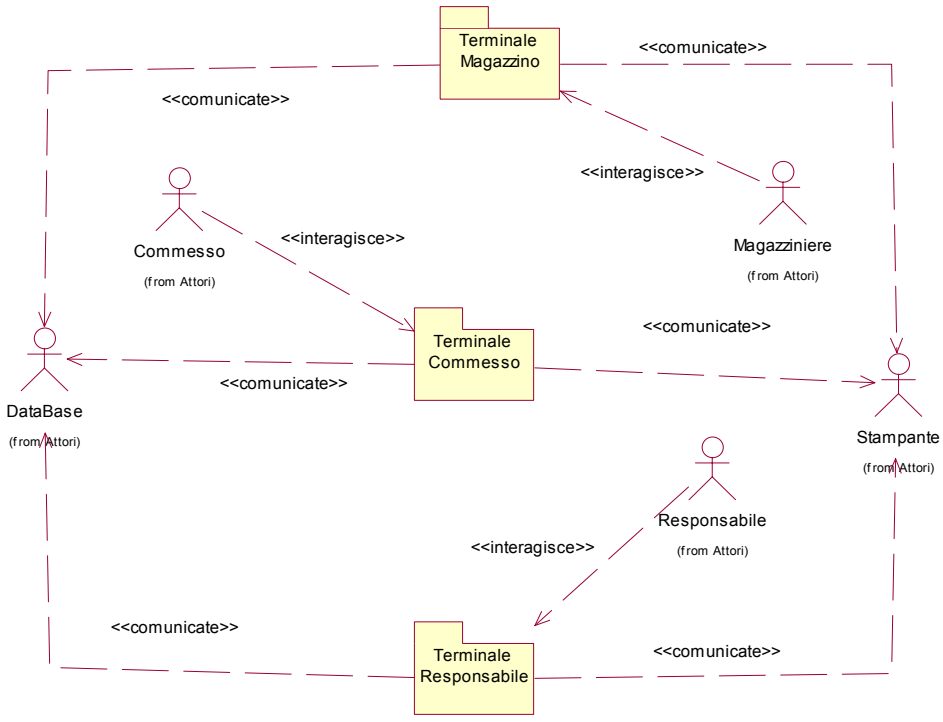
3.5.2.1. Attori

La tabella seguente riporta gli attori che intervengono nell'interazione con il sistema:

Nome	Descrizione
Commesso	Gestisce le vendite dei libri verso i clienti della libreria, consulta la disponibilità di un testo in magazzino, registra le vendite, effettua le prenotazioni.
Magazziniere	Gestisce i movimenti di magazzino, registra le consegne dei libri.
Responsabile	Gestisce gli ordini della libreria, registra e annulla gli ordini.
DataBase	Rappresenta la fonte e il pozzo dei dati del sistema.
Stampante	Consente di ottenere informazioni dal sistema in formato cartaceo per la distribuzione.

3.5.2.2. Use Case

Il diagramma seguente fornisce una vista di insieme del progetto:



I diagrammi dei casi d'uso che seguono forniscono una panoramica delle funzionalità offerte dal software per ogni singola postazione:

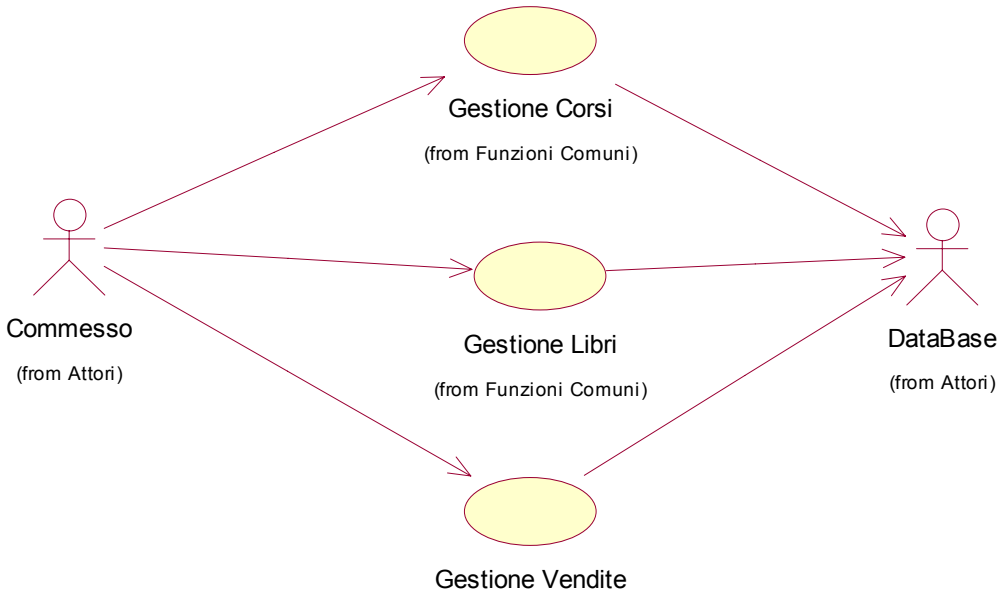


Figura 1 – Posta zione Commesso

Use Case Name	**Gestione Corsi**
Descrizione	Consente all'utente di gestire i vari corsi universitari in termini di inserimento, modifica, eliminazione ed associazione dei libri adottati.

Use Case Name	**Gestione Libri**
Descrizione	Consente all'utente di gestire i vari libri in termini di inserimento, modifica, eliminazione ed associazione ai corsi che li adottano.

Use Case Name	**Gestione Vendite**
Descrizione	Consente all'utente di eseguire tutte le operazioni tipiche di un commesso in una libreria, operazioni che consistono nella ricerca di libri adottati da eventuali corsi, effettuazione di vendite e prenotazioni di libri, ricerca di prenotazioni precedentemente effettuate nonché operazioni di stampa dei vari documenti.

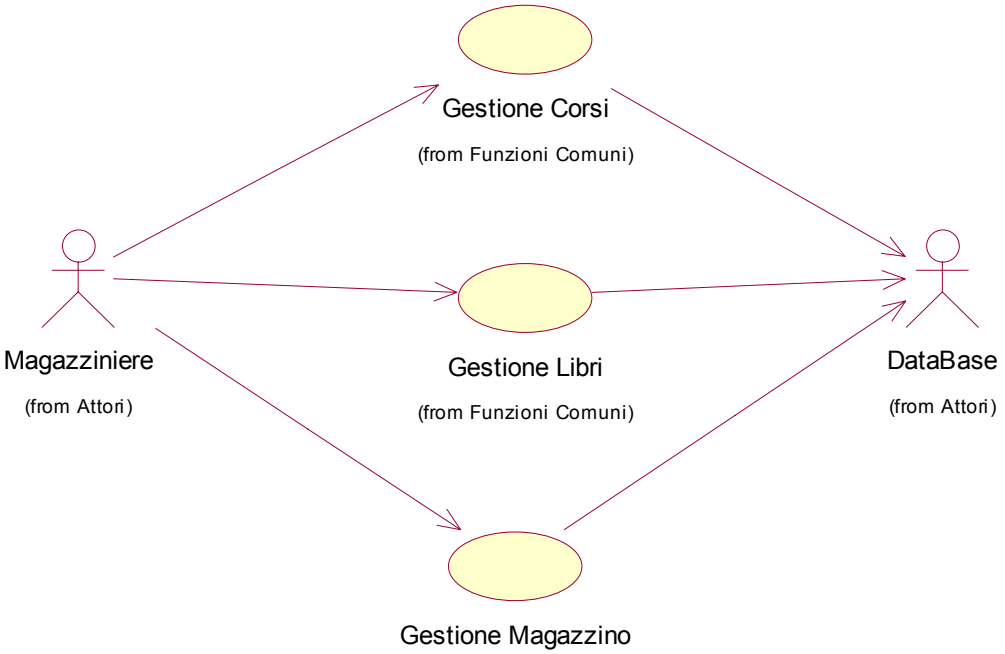


Figura 2 - Postazione Magazzino

Use Case Name	**Gestione Magazzino**
Descrizione	Consente all'utente di eseguire tutte le operazioni tipiche di un magazziniere in una libreria, operazioni che consistono nella ricerca di ordini precedentemente effettuati e controllo delle consegne dei libri ad essi relativi.

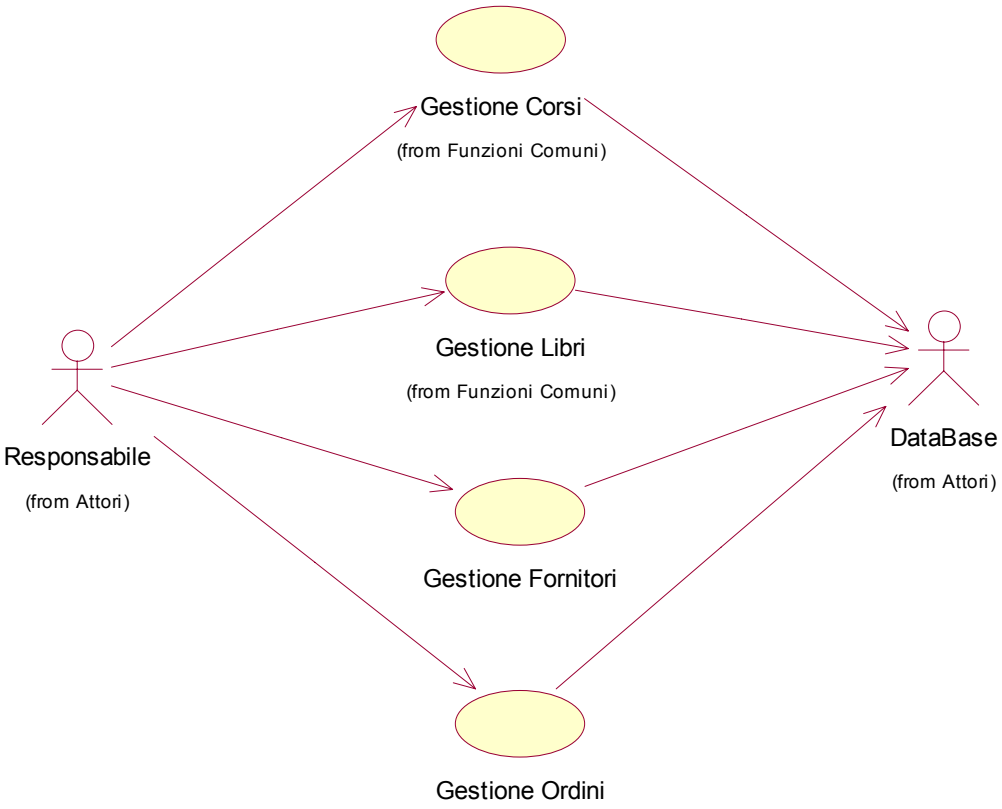


Figura 3 - Postazione Responsabile

Use Case Name	**Gestione Fornitori**
Descrizione	Consente all'utente di gestire i vari fornitori in termini di inserimento, modifica, eliminazione ed associazione agli editori che distribuiscono.

Di seguito si analizzano più in dettaglio le funzionalità descritte precedentemente:

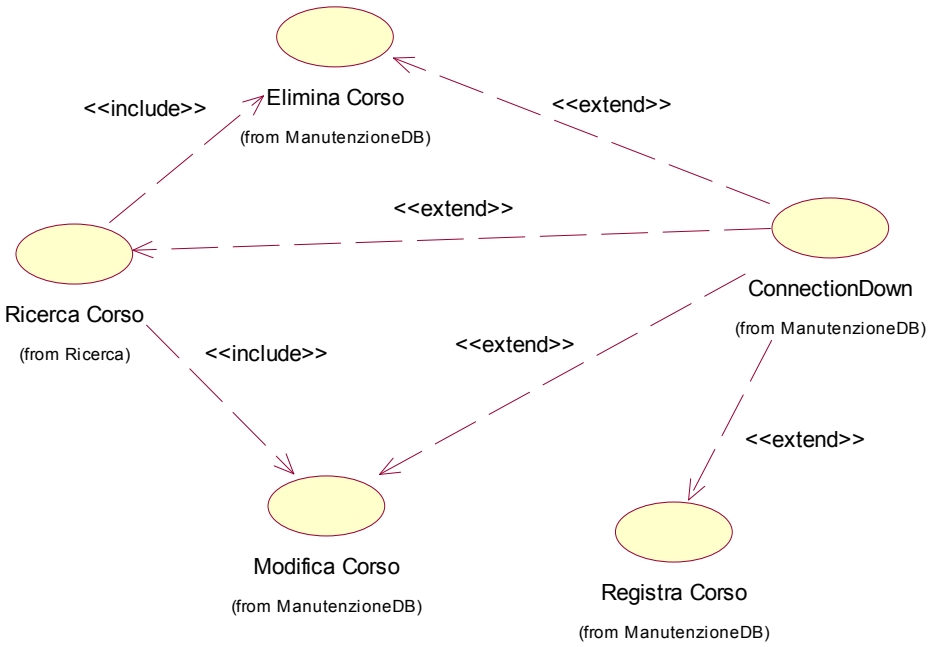


Figura 4 - use case Gestione Corsi

Use Case Name	**Ricerca Corso**
Partecipating Actor	Commesso, DataBase
Descrizione	Consente ad un cliente di chiedere informazioni su uno o più corsi.
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. Cliente fornisce i *dettagli* del *corso* da cercare; 2. Commesso immette tali dati nella *form* relativa alla *ricerca* di un corso e conferma; *Se i dati del corso devono essere aggiornati, viene usato lo use case Modifica Corso.* *Se un corso deve essere eliminato dal database, viene usato lo use case Elimina Corso.*
Exit Condition	BOOKBASE fornisce i *risultati* circa i corsi e gli eventuali libri adottati.

Use Case Name	**Registra Corso**
Partecipating Actor	Commesso, DataBase
Descrizione	Consente al commesso di inserire nel database i dati relativi a nuovi corsi
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. Il commesso attiva la funzionalità "Registra Corso" dal suo terminale; 2. BookBase risponde presentando una *form* al commesso; la form include i campi relativi a *materia*, *docente*, *modulo*, *facoltà*, *indirizzo*, *numero studenti*, *data di inizio* e *data di fine* del *corso*; 3. Il Commesso riempie il form specificando ogni campo; completato il form, il commesso conferma l'inserimento attraverso la pressione del *pulsante* "Salva Corso"
Exit Condition	Inserimento completato

Use Case Name	**Modifica Corso**
Partecipating Actor	Commesso, DataBase
Descrizione	Consente al commesso di modificare i dati relativi ai corsi
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. BookBase sta visualizzando i dati relativi ad una precedente ricerca effettuata dal commesso; 2. Il commesso seleziona il *corso* da modificare e pressa il *pulsante* "Modifica Corso"; 3. BookBase modifica il corso e provvede ad aggiornare il database.
Exit Condition	Modifica completata

Use Case Name	**Elimina Corso**
Partecipating Actor	Commesso, DataBase
Descrizione	Consente al commesso di eliminare un corso dal database
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. BookBase stà visualizzando i dati relativi ad una precedente ricerca effettuata dal commesso; 2. Il commesso seleziona il corso da eliminare dal database e pressa il *pulsante* "Elimina Corso"; 3. BookBase elimina il corso e provvede ad aggiornare il database.
Exit Condition	Eliminazione completata

Use Case Name	**ConnectionDown**
Partecipating Actor	Commesso DataBase
Descrizione	Gestisce l'eventuale interruzione della connessione al Database.
Entry Condition	Connessione al Database interrotta.
Flow of Events	*Lo use case ConnectionDown estende Ricerca Corso, Elimina Corso, Modifica Corso e Registra Corso quando la connessione tra il Commesso ed il DataBase è interrotta.*
Exit Condition	La connessione al DataBase è ristabilita.

Nota: lo use case *Gestione Corsi* del Magazziniere e del Responsabile è identico allo use case del Commesso; si distinguono soltanto per gli attori partecipanti.

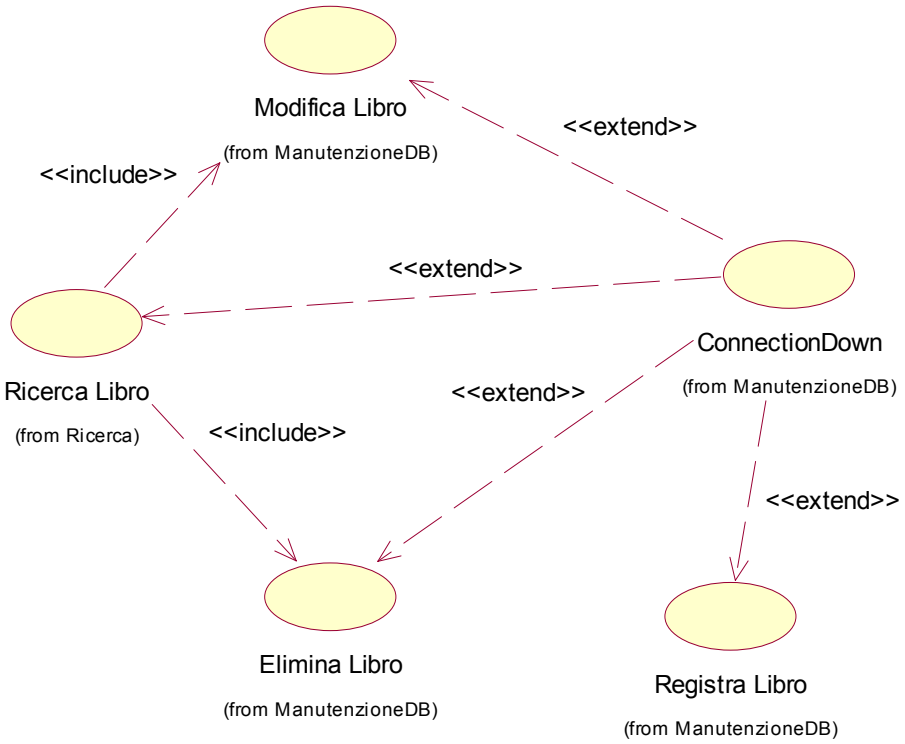


Figura 5 - use case Gestione Libri

Use Case Name	**Ricerca Libro**
Partecipating Actor	Commesso, DataBase
Descrizione	Consente ad un cliente di chiedere informazioni su uno o più libri. Consente inoltre al commesso di verificare l'eventuale disponibilità di quel libro
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. Cliente fornisce i *dettagli* del *libro* da cercare; 2. Commesso immette tali dati nella *form* relativa alla *ricerca* di un testo e conferma; *Se i dati del libro devono essere aggiornati, viene usato lo use case Modifica Libro.* *Se un libro deve essere eliminato dal database, viene usato lo use case Elimina Libro.*
Exit Condition	BOOKBASE fornisce i *risultati* circa i libri e l'eventuale disponibilità in magazzino;

Use Case Name	**Registra Libro**
Partecipating Actor	Commesso, DataBase
Descrizione	Consente al commesso di inserire nel database i dati relativi a nuovi libri
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. Il commesso attiva la funzionalità "Registra Libro" dal suo terminale 2. BookBase risponde presentando una *form* al commesso; la form include i campi relativi a *titolo*, *autori*, *editore*, *ISBN*, *prezzo*, *disponibilità*, *anno di pubblicazione*; 3. Il Commesso riempie il form specificando ogni campo; completato il form, il commesso conferma l'inserimento attraverso la pressione del *pulsante* "Salva Libro"
Exit Condition	Inserimento completato

Use Case Name	**Modifica Libro**
Partecipating Actor	Commesso, DataBase
Descrizione	Consente al commesso di modificare i dati relativi ai libri
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. BookBase sta visualizzando i dati relativi ad una precedente ricerca effettuata dal commesso; 2. Il commesso seleziona il *libro* da modificare dal database, modifica uno o più campi del libro e pressa il *pulsante* "Modifica Libro"; 3. BookBase modifica il libro e provvede ad aggiornare il database.
Exit Condition	Modifica completata

Use Case Name	**Elimina Libro**
Partecipating Actor	Commesso, DataBase
Descrizione	Consente al commesso di eliminare un libro dal database
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. BookBase sta visualizzando i dati relativi ad una precedente ricerca effettuata dal magazziniere; 2. Il commesso seleziona il libro da eliminare dal database e pressa il *pulsante* "Elimina Libro"; 3. BookBase elimina il libro e provvede ad aggiornare il database.
Exit Condition	Eliminazione completata

Use Case Name	**ConnectionDown**
Partecipating Actor	Commesso DataBase
Descrizione	Gestisce l'eventuale interruzione della connessione al Database.
Entry Condition	Connessione al Database interrotta.
Flow of Events	*Lo use case ConnectionDown estende Ricerca Libro, Elimina Libro, Modifica Libro e Registra Libro quando la connessione tra il Commesso ed il DataBase è interrotta.*
Exit Condition	La connessione al DataBase è ristabilita.

Nota: lo use case *Gestione Libri* del Magazziniere e del Responsabile è identico allo use case del Commesso; si distinguono soltanto per gli attori partecipanti.

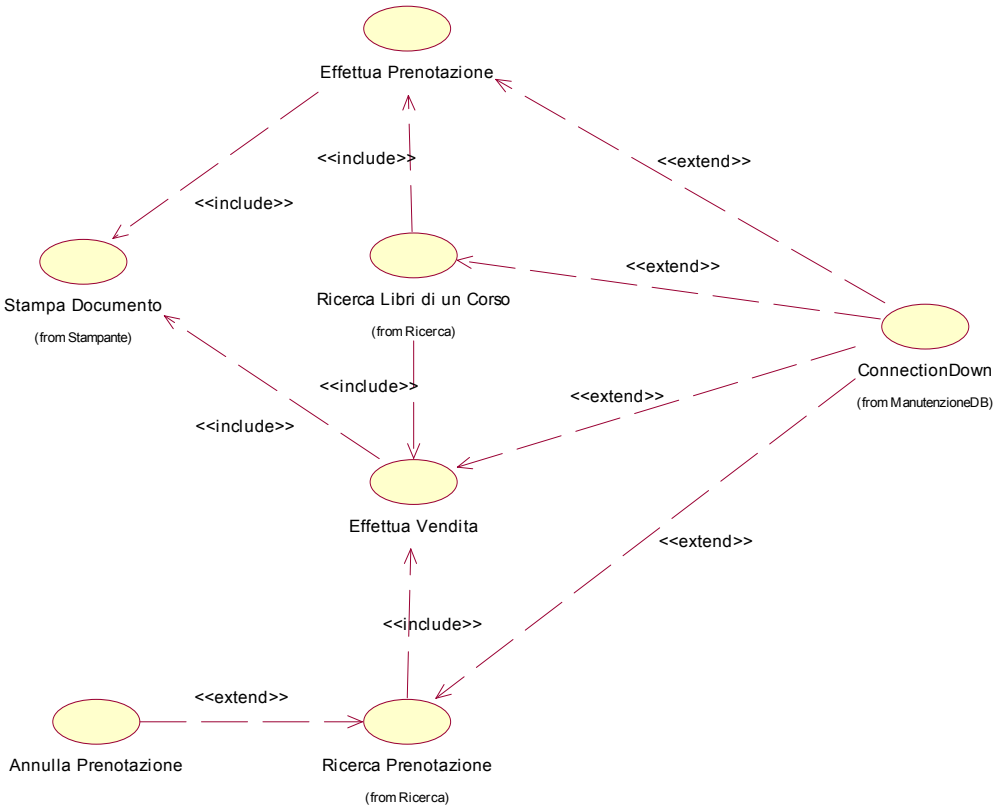


Figura 6 - use case Gestione Vendite

Use Case Name	**Ricerca Libro Di Un Corso**
Partecipating Actor	Commesso, DataBase, Cliente
Descrizione	Consente ad un cliente di chiedere informazioni su uno o più libri adottati da un particolare corso.
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. Cliente fornisce alcune *informazioni* circa il libro e/o il corso da cercare; 2. Commesso immette tali dati nella *form* relativa e conferma; 3. BOOKBASE fornisce il dettaglio dei *risultati* e l'eventuale *disponibilità* in magazzino; 4. Il commesso comunica tali dati al cliente. *Se il cliente chiede l'acquisto del libro di cui si è effettuata la ricerca, viene invocato lo use case Effettua Vendita.* *Se il cliente chiede la prenotazione del libro di cui si è effettuata la ricerca, viene invocato lo use case Effettua Prenotazione.*
Exit Condition	Consultazione completata

Use Case Name	**Ricerca Prenotazione**
Partecipating Actor	Commesso, DataBase, Cliente
Descrizione	Consente al commesso di cercare i dati relativi alla prenotazione effettuata precedentemente al cliente
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. il Cliente fornisce il *documento di prenotazione* 2. il Commesso apre la *form*, immette i dati e conferma pressando il *pulsante* "Ricerca Prenotazione"; 3. BOOKBASE fornisce al commesso un *report* contenente delle prenotazioni; 4. Il commesso seleziona la prenotazione di interesse. *Se il cliente conferma i dati del documento di prenotazione, viene invocato lo use case Effettua Vendita.*
Exit Condition	BookBase visualizza dettagli della prenotazione nella *form di prenotazione*;

Use Case Name	**Effettua Vendita**
Partecipating Actor	Commesso, DataBase, Cliente
Descrizione	Consente ad un cliente di acquistare uno o più libri, comprende le operazioni riguardanti la verifica dei dati relativi al libro. Al termine della vendita sarà fornito al cliente anche un documento di vendita
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. Il Cliente richiede la *vendita* di alcuni *libri*; 2. Il commesso verifica la disponibilità dei libri 3. Il commesso comunica il prezzo 4. il Cliente fornisce il denaro 5. Il commesso immette i dati riguardanti i libri nella apposita *form di vendita* e salva le informazioni nel sistema pressando il *pulsante* "Salva Vendita"; 6. Il commesso fornisce i libri ed il *documento di vendita* 7. il Commesso stampa il *documento di vendita*. *Quando il commesso pressa il pulsante "stampa" verrà invocato lo use case Stampa Documento.*
Exit Condition	Stampa documento di vendita

Use Case Name	**Effettua Prenotazione**
Partecipating Actor	Commesso, DataBase, Cliente
Descrizione	Consente ad un cliente di richiedere la prenotazione di un libro previo il rilascio di un acconto e del nominativo. Al termine della prenotazione sarà fornito al cliente un documento che attesta l'avvenuta prenotazione contenente tutti i dati di interesse
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	1. il Cliente richiede la *prenotazione* di un *libro*; 2. il Commesso richiede l'acconto e il nominativo della prenotazione; 3. il Cliente fornisce il denaro e le informazioni necessarie; 4. Il Commesso immette i dati riguardanti il libro nella apposita *form di prenotazione* e salva le informazioni nel sistema pressando il *pulsante* "Salva Prenotazione"; 5. il Commesso stampa il *documento di prenotazione*. *Quando il commesso clicca sul pulsante "stampa" verrà invocato lo use case Stampa Documento*
Exit Condition	Stampa documento di prenotazione

Use Case Name	**Annulla Prenotazione**
Partecipating Actor	Commesso, DataBase Cliente
Descrizione	Consente al commesso di annullare una prenotazione
Entry Condition	Commesso disponibile Postazione vendite operativa
Flow of Events	*Lo use case Annulla Prenotazione estende Ricerca Prenotazione quando il commesso ha la necessità di annullare una prenotazione verso un cliente.* 1. Il Cliente richiede di annullare la prenotazione e consegna il documento di prenotazione rilasciato precedentemente 2. Il commesso apre la form di ricerca prenotazione e inserisce i dati della prenotazione 3. Il commesso individua la prenotazione 4. Il commesso pressa il *pulsante* "Annulla Prenotazione" e conferma
Exit Condition	Operazione confermata

Use Case Name	**Stampa Documento**
Partecipating Actor	Commesso, Stampante
Descrizione	Provvede alla stampa dei dati visualizzati su schermo.
Entry Condition	Postazione vendite operativa Stampare on-line
Exit Condition	Stampa completata

Use Case Name	**ConnectionDown**
Partecipating Actor	Commesso DataBase
Descrizione	Gestisce l'eventuale interruzione della connessione al Database.
Entry Condition	Connessione al Database interrotta.
Flow of Events	*Lo use case ConnectionDown estende Ricerca Prenotazione, Effettua Vendita, Effettua Prenotazione e Ricerca Libri Di Un Corso quando la connessione tra il Commesso ed il DataBase è interrotta.*
Exit Condition	La connessione al DataBase è ristabilita.

Nota: il Cliente è stato considerato come un attore che interagisce con il Commesso, e ragionevolmente esso non interagirà con BOOKBASE. E' per tale motivo che il ruolo non è stato inserito nel paragrafo [*3.5.2.1 Attori*].

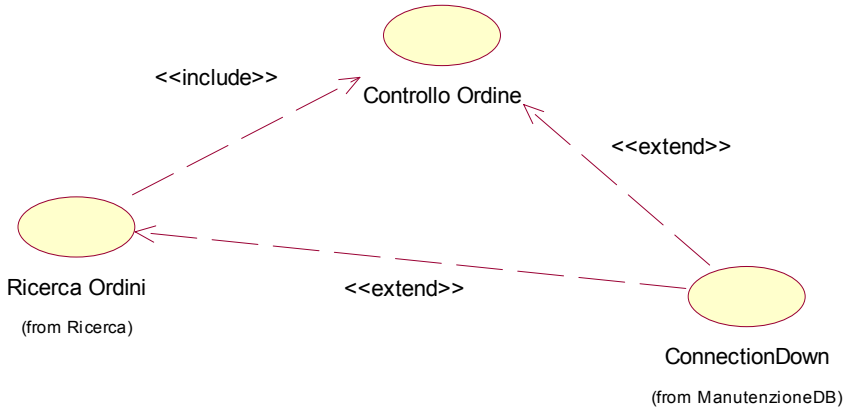


Figura 7 - use case Gestione Magazzino

Use Case Name	**Ricerca Ordini**
Participating Actor	Magazziniere, DataBase.
Descrizione	Permette la ricerca di tutti gli ordini ancora non evasi accedendo al database relativo agli ordini.
Entry Condition	Postazione magazzino operativa.
Flow of Events	1. Il magazziniere riempie i campi relativi all'*ordine* da ricercare,nella relativa *form*; 2. Il magazziniere conferma la ricerca pressando il *pulsante* "Ricerca Ordine"; 3. BookBase visualizza un *report* contenente degli ordini; 4. Il magazziniere seleziona l'ordine opportuno e visualizza i dettagli nella *form di ordine.* *Se della merce è arrivata e deve essere inserita nel magazzino, viene invocato lo use case Controllo Ordine*
Exit Condition	Ricerca completata

Use Case Name	**Controllo Ordine**
Partecipating Actor	Magazziniere, DataBase.
Descrizione	Consente all'addetto al magazzino di aggiornare le giacenze degli articoli ivi presenti. Al termine di un aggiornamento degli articoli in fornitura verrà anche aggiornato l'archivio relativo agli ordini pendenti che potrebbero diventare ordini evasi.
Entry Condition	Arrivo merce, Magazziniere disponibile, Postazione magazzino operativa.
Flow of Events	1. Il magazziniere controlla se i *libri* richiesti nell'*ordine* sono stati consegnati e in che quantità. 2. Il magazziniere carica i libri in arrivo nella *form* apposita per l'ordine e inserisce le relative quantità. 3. conferma l'immissione attraverso il *pulsante* apposito per l'aggiornamento dell'ordine.
Exit Condition	Articoli in arrivo caricati nel database.

Use Case Name	**ConnectionDown**
Partecipating Actor	Magazzino DataBase
Descrizione	Gestisce l'eventuale interruzione della connessione al Database.
Entry Condition	Connessione al Database interrotta.
Flow of Events	*Lo use case ConnectionDown estende Ricerca Ordini e Controllo Ordine quando la connessione tra il Magazziniere ed il DataBase è interrotta.*
Exit Condition	La connessione al DataBase è ristabilita.

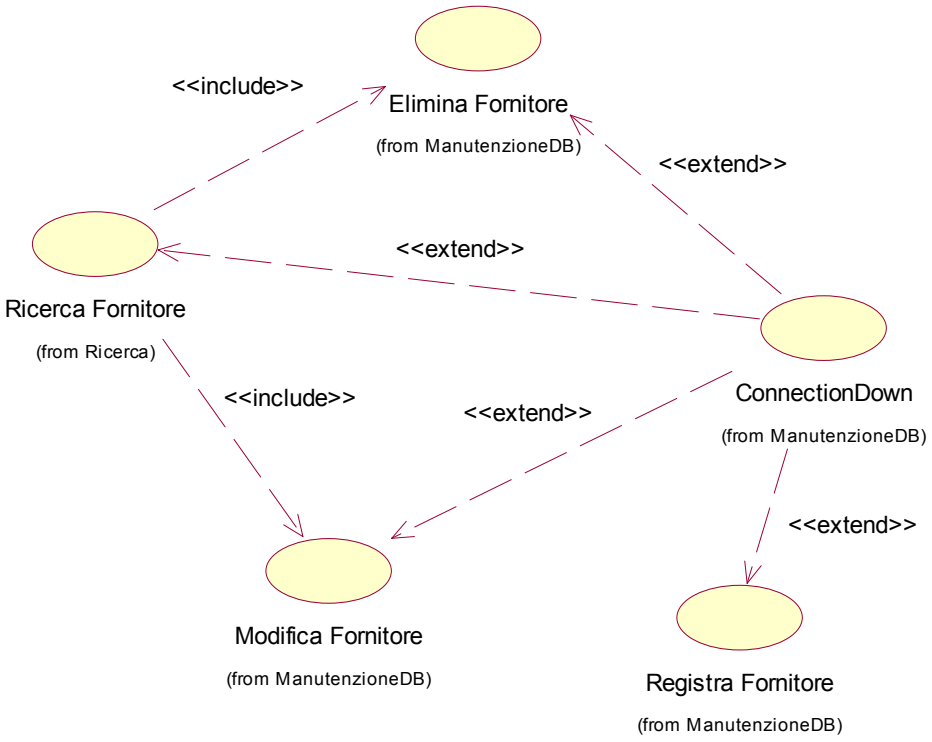


Figura 8 - use case Gestione Fornitori

Use Case Name	**Ricerca Fornitore**
Participating Actor	Responsabile, DataBase
Descrizione	Permette di cercare informazioni relative a uno o più fornitori.
Entry Condition	Postazione Responsabile operativa Il responsabile ricerca informazioni.
Flow of Events	1. Responsabile accede alla *form* relativa 2. Il responsabile inserisce, nei campi appositi, tutte le informazioni richieste; 3. Bookbase raccoglie le informazioni immesse e, in base ai filtri impostati effettua la ricerca; 4. BookBase visualizza i *risultati* della ricerca. *Se il responsabile ritiene necessario di modificare alcuni dati relativi a un fornitore, viene invocato lo use case Modifica Fornitore.* *Se il responsabile ritiene necessario eliminare un fornitore, viene invocato lo use case Elimina Fornitore.*
Exit Condition	Ricerca completata

Use Case Name	**Registra Fornitore**
Participating Actor	Responsabile, DataBase
Descrizione	Permette di inserire un nuovo insieme di informazioni relative a un fornitore
Entry Condition	Postazione Responsabile operativa Il responsabile vuole aggiornare nel database alcuni dati utili per gestione dell'esercizio.
Flow of Events	1. Responsabile accede alla *form* relativa 2. Il responsabile inserisce, nei campi appositi, tutte le informazioni richieste; 3. Il responsabile conferma le immissioni e il sistema provvede ad aggiornare il database.
Exit Condition	Aggiornamento database completato

Use Case Name	**Elimina Fornitore**
Partecipating Actor	Responsabile, DataBase
Descrizione	Consente al responsabile di eliminare un fornitore dal database
Entry Condition	Postazione responsabile operativa
Flow of Events	1. BookBase sta visualizzando i dati relativi ad una precedente ricerca effettuata dal responsabile; 2. Il responsabile seleziona il fornitore da eliminare dal database e clicca sul *pulsante* "Elimina Fornitore"; 3. BookBase elimina il fornitore e provvede ad aggiornare il database.
Exit Condition	Eliminazione completata

Use Case Name	**Modifica Fornitore**
Partecipating Actor	Responsabile, DataBase
Descrizione	Consente al responsabile di modificare i dati relativi ai fornitori
Entry Condition	Postazione responsabile operativa
Flow of Events	1. BookBase sta visualizzando i dati relativi ad una precedente ricerca effettuata dal responsabile; 2. Il responsabile seleziona il fornitore da modificare dal database e clicca sul *pulsante* "Modifica Fornitore"; 3. BookBase modifica i dati del fornitore e provvede ad aggiornare il database.
Exit Condition	Modifica completata

Use Case Name	**ConnectionDown**
Partecipating Actor	Responsabile DataBase
Descrizione	Gestisce l'eventuale interruzione della connessione al Database.
Entry Condition	Connessione al Database interrotta.
Flow of Events	*Lo use case ConnectionDown estende Ricerca Fornitori, Registra Fornitore, Modifica Fornitore, Elimina Fornitore quando la connessione tra il Responsabile ed il DataBase è interrotta.*
Exit Condition	La connessione al DataBase è ristabilita.

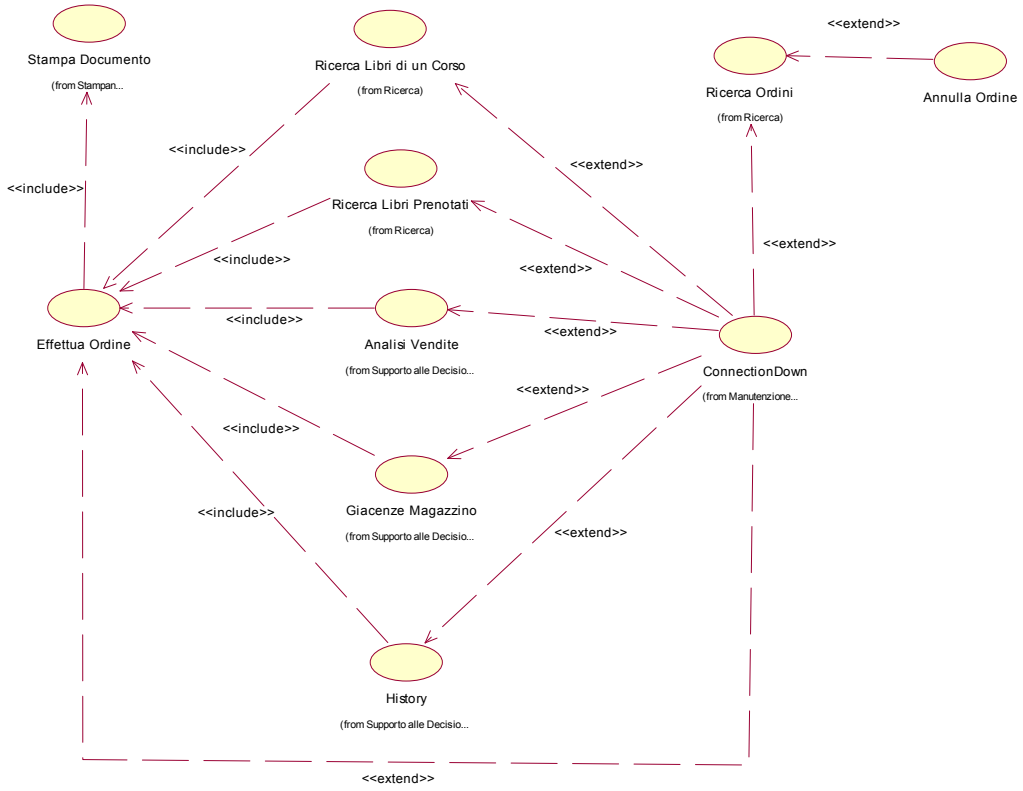


Figura 9 - use case Gestione Ordini

Use Case Name	**Analisi Vendite**
Participating Actor	Responsabile, DataBase
Descrizione	Permette l'estrazione, dalla base di dati di informazioni relative all'aumento di efficienza e produttività dell'attività
Entry Condition	Postazione Responsabile operativa Il responsabile cerca informazioni utili dell'attività
Flow of Events	1. Responsabile accede alla *form* relativa all'analisi vendite; 2. BookBase interroga il DB in base ai filtri impostati dal responsabile (intervallo temporale di analisi); 3. BookBase visualizza un *report* contenente *libri* caratterizzati da: *titolo*, *editore*, *prezzo di vendita*, *numero di copie vendute* e *percentuale quantità venduta/giacenza* di magazzino. *Se il responsabile ritiene necessario, sulla base dei dati visualizzati da BookBase, effettuare l'ordine di qualche libro, viene invocato lo use case Effettua Ordine.*
Exit Condition	Fine della consultazione database

Use Case Name	**History**
Participating Actor	Responsabile, DataBase
Descrizione	Permette l'estrazione, dalla base di dati di informazioni relative all'aumento di efficienza e produttività dell'attività
Entry Condition	Postazione Responsabile operativa Il responsabile cerca informazioni utili dell'attività
Flow of Events	1. Responsabile accede alla *form* relativa alla storia acquisti/vendite dell'esercizio in corso; 2. BookBase interroga il DB in base ai filtri impostati (intervallo temporale di analisi) dal responsabile 3. BookBase visualizza un *report* contenente i libri caratterizzati da: quantità venduta e acquistata e la data dell'ultima vendita, in riferimento al periodo in questione; *Se il responsabile ritiene necessario, sulla base dei dati visualizzati da BookBase, effettuare l'ordine di qualche libro, viene invocato lo use case Effettua Ordine.*
Exit Condition	Fine della consultazione database

Use Case Name	**Giacenze Magazzino**
Participating Actor	Responsabile
Descrizione	Permette l'estrazione, dalla base di dati di informazioni relative all'aumento di efficienza e produttività dell'attività
Entry Condition	Postazione Responsabile operativa Il responsabile cerca informazioni utili dell'attività
Flow of Events	1. Responsabile accede alla *form* relativa alle giacenze di magazzino dell'esercizio in corso; 2. BookBase interroga il DB 3. BookBase visualizza un *report* contenente i libri caratterizzati dalla quantità presente in magazzino, con in evidenza le giacenze che sono al di sotto di una soglia prestabilita. *Se il responsabile ritiene necessario, sulla base dei dati visualizzati da BookBase, effettuare l'ordine di qualche libro, viene invocato lo use case Effettua Ordine.*
Exit Condition	Fine della consultazione database

Use Case Name	**Ricerca Libri Prenotati**
Partecipating Actor	Responsabile, DataBase
Descrizione	Consente al commesso di cercare le prenotazioni pendenti al fine di inoltrare i rispettivi ordini
Entry Condition	Postazione responsabile operativa
Flow of Events	1. il responsabile apre la *form*, immette i dati e conferma pressando il *pulsante* "Ricerca Prenotazione"; 2. BOOKBASE fornisce al responsabile un *report* contenente delle prenotazioni pendenti, con in dettaglio l'insieme dei libri prenotati e in quale quantità; *Se il responsabile, in base ai dati visualizzati, ritiene necessario effettuare un ordine, viene invocato lo use case Effettua Ordine.*
Exit Condition	Consultazione completata

Use Case Name	**Effettua Ordine**
Participating Actor	Responsabile, DataBase
Descrizione	Consente di effettuare ordini presso i fornitori. Gli ordini verranno effettuati in base alle prenotazioni ed alle giacenze di magazzino, avvalendosi di un supporto decisionale sulla base di informazioni inserite nel database
Entry Condition	Postazione Responsabile operativa Prenotazione libri pendente
Flow of Events	1. BookBase sta visualizzando il riepilogo degli ordini da eseguire, creato in precedenza dal responsabile 2. il responsabile effettua l'ordine dei libri fissando le quantità da ordinare per ogni libro e specificando i dati del fornitore 3. il responsabile conferma l'ordine pressando il *pulsante* "Salva Ordine"; 4. il responsabile stampa il *documento di ordine*. *Quando il responsabile clicca sul pulsante "stampa" verrà invocato lo use case Stampa Documento.*
Exit Condition	Ordine effettuato

Use Case Name	**Annulla Ordine**
Participating Actor	Responsabile, DataBase
Descrizione	Consente al Responsabile di annullare un ordine
Entry Condition	Postazione Responsabile operativa Il responsabile vuole annullare un ordine ancora pendente
Flow of Events	*Lo use case Annulla Ordine estende lo use case RicercaOrdine quando il responsabile vuole annullare un ordine ancora pendente.* 1. BookBase sta visualizzando il risultato della ricerca, effettuata in precedenza dal responsabile; 2. Il responsabile seleziona un ordine e clicca sul *pulsante* "Annulla Ordine"; 3. BookBase provvederà a cancellare l'ordine selezionato, aggiornando i dati del database.
Exit Condition	Ordine cancellato Database aggiornato

Use Case Name	**Ricerca Libro Di Un Corso**
Partecipating Actor	Responsabile, DataBase
Descrizione	Consente al responsabile di ricercare informazioni su uno o più libri adottati da un particolare corso.
Entry Condition	Postazione responsabile operativa
Flow of Events	1. Responsabile immette tali dati di ricerca nella *form* relativa e conferma; 2. BOOKBASE fornisce il dettaglio dei *risultati*; *Se il responsabile, in base ai dati visualizzati, ritiene necessario effettuare un ordine, viene invocato lo use case Effettua Ordine.*
Exit Condition	Consultazione completata

Use Case Name	**Ricerca Ordini**
Participating Actor	Responsabile, DataBase
Descrizione	Permette la ricerca di tutti gli ordini ancora non evasi accedendo al database relativo agli ordini
Entry Condition	Postazione Responsabile operativa Il responsabile vuole sollecitare qualche ordine, o controllare lo stato degli ordini in arrivo
Flow of Events	1. Responsabile accede alla *form* relativa alla ricerca ordini; 2. Egli immette uno o più campi in tale form, impostando così i criteri di ricerca; 3. BookBase interroga il Database e visualizza il *risultato* della ricerca; *Se il responsabile vuole modificare qualche ordine ancora pendente, viene invocato lo use case Modifica Ordine.*
Exit Condition	Ricerca completata

Use Case Name	**Stampa Documento**
Partecipating Actor	Responsabile, Stampante
Descrizione	Provvede alla stampa dei dati visualizzati su schermo.
Entry Condition	Postazione responsabile operativa Necessità di stampare
Exit Condition	Stampa completata

Use Case Name	**ConnectionDown**
Partecipating Actor	Responsabile DataBase
Descrizione	Gestisce l'eventuale interruzione della connessione al Database.
Entry Condition	Connessione al Database interrotta.
Flow of Events	*Lo use case ConnectionDown estende Ricerca Libri Di Un Corso, Analisi Vendite, History, Ricerca Libri Prenotati, Giacenze Magazzino, Effettua Ordine, Ricerca Ordini quando la connessione tra il Responsabile ed il DataBase è interrotta.*
Exit Condition	La connessione al DataBase è ristabilita.

Si riporta di seguito una visione completa delle funzionalità delle tre postazioni in termini di use case diagram. Per maggiore leggibilità si è omesso lo use case *ConnectionDown:*

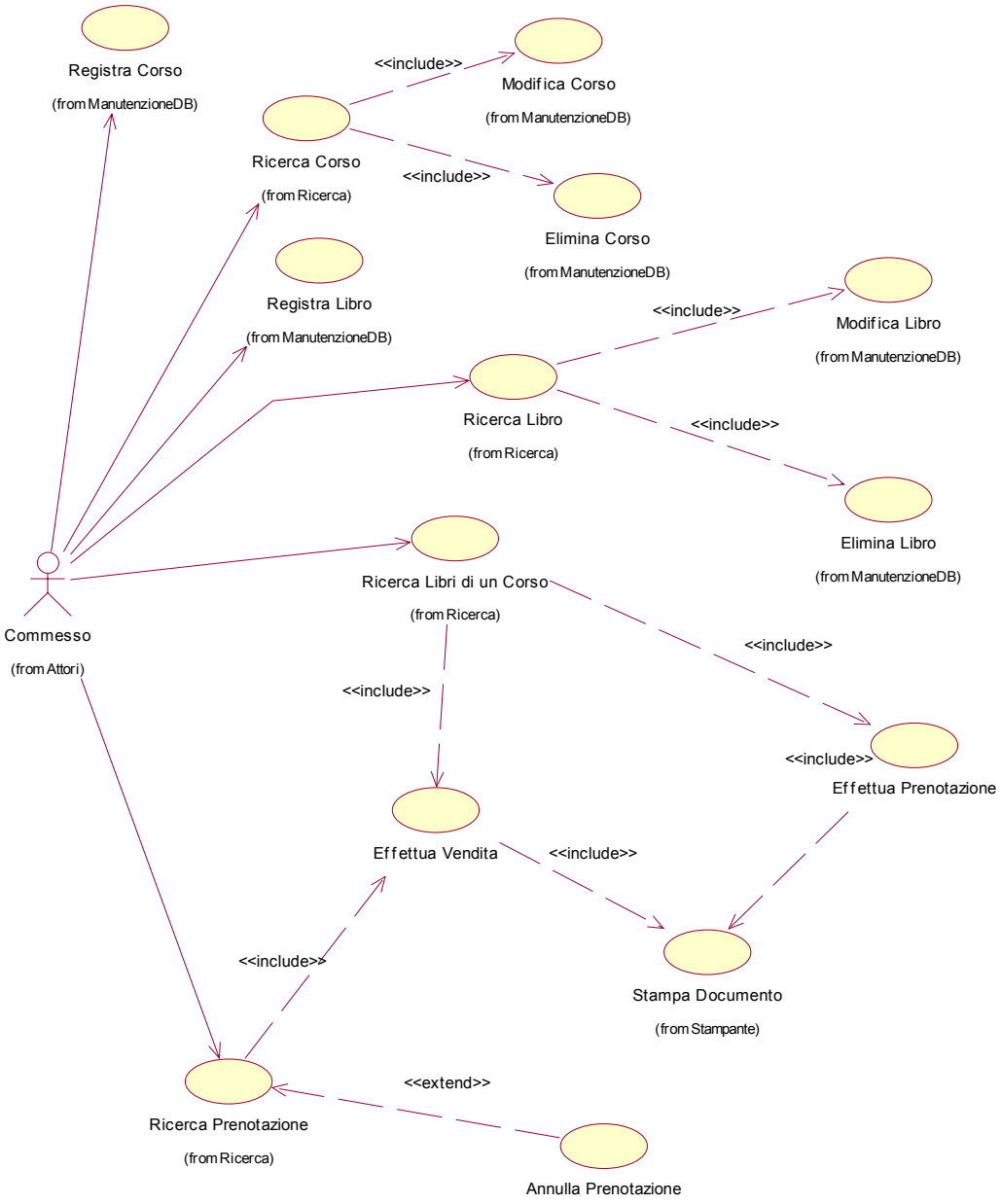


Figura 10 - Postazione Commesso

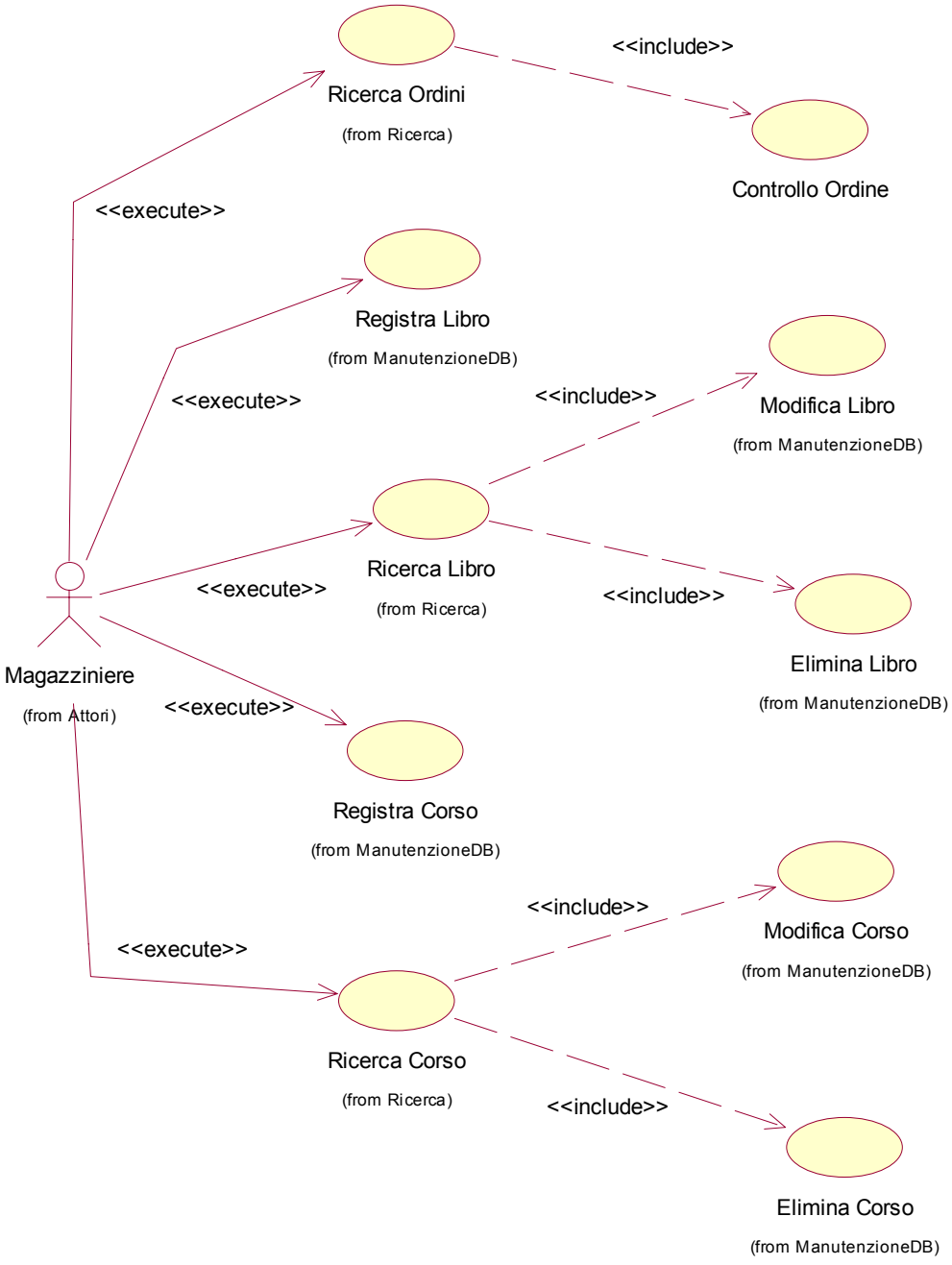


Figura 11 - Postazione Magazzino

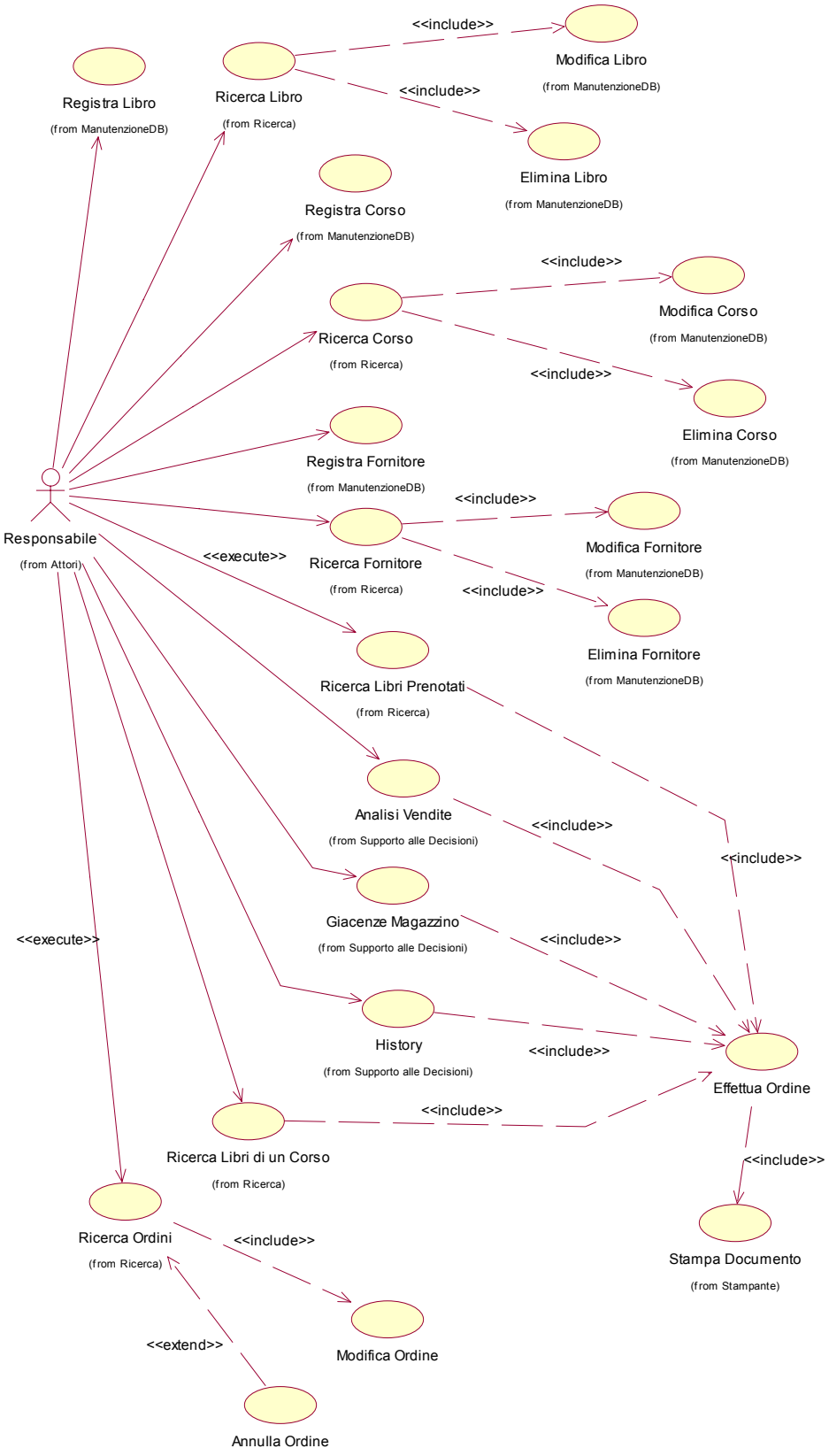


Figura 12 - Postazione Responsabile

3.5.3. Object Models

3.5.3.1. Data Dictionary

L'euristica di Abbot, applicata agli use case sopra descritti, porta all'individuazione dei seguenti oggetti partecipanti:

Entity Object

Nome	Descrizione
Commesso	Gestisce le Vendite dei libri verso i clienti della libreria, consulta la disponibilità di un Libro in magazzino, registra le Vendite, effettua le Prenotazioni.
Magazziniere	Gestisce i movimenti di magazzino, registra le consegne dei libri, determina se un Ordine è evaso o meno.
Responsabile	Gestisce gli Ordini della libreria, registra e annulla gli ordini, utilizza il Supporto alle Decisioni per l'approvvigionamento.
Database	Rappresenta la fonte e il pozzo di dati del sistema.
Stampante	Consente di ottenere informazioni dal sistema in formato cartaceo per la distribuzione.
Libro	Insieme di dati che caratterizzano un testo, ovvero Autore, Titolo, Editore, ISBN, Prezzo, Anno di Pubblicazione, Quantità presente in magazzino.
Prenotazione	Insieme di dati costituita dall'elenco dei Libri prenotati, la Quantità per ogni libro, il Nominativo del cliente, l'Acconto fornito e la Data di prenotazione.
Vendita	E' una entità che tiene conto , oltre alla lista dei Libri venduti, della Data di vendita.
Ordine	Contiene informazione circa la lista dei Libri da ordinare, la Data di effettuazione dell'ordine, ed il nominativo del Fornitore cui l'ordine è destinato.
Corso	Insieme di informazioni relativi ai corsi universitari per i quali la libreria fornisce libri richiesti e consigliati. Un corso è identificato da Materia, Facoltà, Indirizzo, Docente titolare, inizio del modulo, fine del modulo e Numero di studenti per quel corso.
Fornitore	Insieme di informazioni relativi ai fornitori che riforniscono la libreria. Un fornitore è identificato da:Ragione Sociale, Partita IVA, Telefono, Sede, Città e CAP.
LibroPrenotato	E' l'informazione che tiene traccia di un libro prenotato, ereditando quindi tutti gli attributi dell'oggetto libro, e specificandone addizionalmente la quantità.
LibroVenduto	E' l'informazione che tiene traccia di un libro venduto, ereditando tutti gli attributi dell'oggetto libro, e specificandone addizionalmente la quantità.
LibroOrdinato	E' l'informazione che tiene traccia di un libro ordinato, ereditando tutti gli attributi dell'oggetto libro, e specificandone addizionalmente la quantità.

Nome	Descrizione
LibroHistory	E' l'informazione che tiene traccia della storia degli acquisti e delle vendite del libro, ne eredita tutti i suoi attributi, e ne specifica addizionalmente la quantità venduta, quella acquistata.
AnalisiLibroVenduto	E' l'informazione che tiene traccia dell'analisi del venduto di un libro, ne eredita tutti i suoi attributi, e ne specifica addizionalmente la quantità e la percentuale 'quantità venduta / giacenze di magazzino'.
ReportHistory	Tale oggetto risulta composto dall'aggregato di zero o più oggetti di tipo *Libro History* e rappresenta una lista dei libri ciascuno caratterizzato da una quantità venduta e acquistata, in riferimento a un periodo di riferimento. E' il contenuto informativo della form visualizzata al responsabile nello use case *History*.
ReportLibriPrenotati	Tale oggetto risulta composto dall'aggregato di zero o più oggetti di tipo *Libro Prenotato* e rappresenta una lista dei libri prenotati. E' il contenuto informativo della form visualizzata al responsabile nello use case *Ricerca Libri Prenotati*.
ReportPrenotazione	Tale oggetto risulta composto dall'aggregato di zero o più oggetti di tipo *Prenotazione* e rappresenta una lista delle prenotazioni. E' il contenuto informativo della form visualizzata al commesso nello use case *Ricerca Prenotazione*.
ReportLibri	Tale oggetto risulta composto dall'aggregato di zero o più oggetti di tipo *Libro e* rappresenta una lista di libri. E' il contenuto informativo della form visualizzata al commesso nello use case *Ricerca Libri*.
ReportAnalisiVendite	Tale oggetto risulta composto dall'aggregato di zero o più oggetti di tipo *Analisi Libro Venduto*. E' il contenuto informativo della form visualizzata al responsabile nello use case *Analisi Vendite.*
ReportLibriVenduti	Tale oggetto risulta composto dall'aggregato di zero o più oggetti di tipo *Libro Venduto* e rappresenta una lista dei libri venduti. E' il contenuto informativo della form visualizzata al commesso nello use case *Ricerca Vendita*.
ReportCorsi	Tale oggetto risulta composto dall'aggregato di zero o più oggetti di tipo *Corso* e rappresenta una lista dei corsi. E' il contenuto informativo della form visualizzata al responsabile/commesso nello use case *Ricerca Corso*.
ReportOrdini	Tale oggetto risulta composto dall'aggregato di zero o più oggetti di tipo *Ordine* e rappresenta una lista degli ordini. E' il contenuto informativo della form visualizzata al responsabile nello use case *Ricerca Ordini*.
ReportFornitori	Tale oggetto risulta composto dall'aggregato di zero o più oggetti di tipo *Fornitore* e rappresenta una lista dei fornitori. E' il contenuto informativo della form visualizzata al responsabile nello use case *Ricerca Fornitori*.

Boundary Object

Nome	Descrizione
PrenotazioneForm	Form usata per raccogliere i dati in ingresso della *Prenotazione*. Tale form è presentata al Commesso nella *Postazione Vendite* quando viene attivata la funzione "Prenotazione". La *PrenotazioneForm* contiene i campi per specificare tutti gli attributi tipici di una prenotazione, un meccanismo per accodare successive *Prenotazioni* e un pulsante di conferma quando la form è completata.
RicercaLibroForm	Form che viene usata per effettuare una ricerca di un *Libro*; questa visualizza un insieme di informazione di riepilogo su un particolare libro come titolo, autori, editore, ISBN etc; consente inoltre di modificare ed eliminare un libro.
VenditaForm	Form usata per raccogliere i dati in ingresso della *Vendita*. Tale form è presentata al Commesso nella *Postazione Vendite* quando viene attivata la funzione "Vendita".
OrdineForm	Viene visualizzata al *Magazziniere* (*Responsabile)* nella *Postazione Magazzino (Postazione Responsabile)* ed è una Form che riassume il contenuto di un *Ordine*, ovvero l'elenco degli articoli e delle rispettive quantità costituenti l'ordine stesso. Tale form è presentata al *Responsabile* nella *Postazione Responsabile* quando viene attivata la funzione "Ordine". *OrdineForm* contiene i campi per specificare tutti gli attributi tipici di una ordinazione, un meccanismo per accodare successivi *Libri* e un pulsante di conferma quando la form è completata.
RicercaOrdiniForm	Form che viene usata dal *Magazziniere/Responsabile* per effettuare una ricerca di un *Ordine*. Il *Magazziniere/Responsabile* inserisce dati su alcuni campi della form e il sistema, effettuata la ricerca, completa i campi mancanti.
GiacenzeMagazzinoForm	Visualizza l'elenco dei libri in magazzino la cui quantità risulta al di sotto di una certa soglia. Il *Responsabile* ha una visione sempre aggiornata di quali libri sono in esaurimento e decidere se effettuare qualche ordine.
RicercaLibriPrenotatiForm	Visualizza l'elenco delle prenotazioni di libri il cui relativo ordine è ancora da effettuare. Il *Responsabile* ha una visione sempre aggiornata delle prenotazioni attese e può decidere se effettuare qualche ordine.
HistoryForm	Form che visualizza un riepilogo sulla storia delle vendita e degli acquisti effettuati in un particolare intervallo temporale. E' possibile visualizzare la storia relativi agli acquisti-vendite per tutti i libri oppure specificando un particolare libro.
AnalisiVenditeForm	Form che visualizza un riepilogo sulle vendite effettuate in un particolare intervallo temporale e la percentuale data dal rapporto QV/G.
RegistraCorsoForm	La Form consente di inserire i dati relativi ad un nuovo corso nel database.

Nome	Descrizione
RegistraLibroForm	La Form consente di inserire i dati relativi ad un nuovo libro nel database.
RegistraFornitoreForm	La Form consente di inserire i dati relativi ad un nuovo fornitore nel database.
RicercaLibriDiCorsoForm	Form che viene usata dal *Commesso/Responsabile* per effettuare la ricerca di una *Libri* associati ad un particolare *Corso*. Il *Commesso/Responsabile* inserisce dati su alcuni campi della form e il sistema, effettuata la ricerca, completa i campi mancanti.
RicercaCorsoForm	Form che visualizza un insieme di informazione di riepilogo su un particolare corso universitario come libri richiesti, consigliati, numero di studenti per quel corso, etc; consente inoltre di modificare ed eliminare un corso.
RicercaFornitoriForm	Form che visualizza un insieme di informazione di riepilogo su un particolare fornitore come ragione sociale, sede, partita IVA, etc; consente inoltre di modificare ed eliminare un fornitore.
RicercaPrenotazioniForm	Form che viene usata dal *Commesso* per effettuare la ricerca di una *Prenotazione*. Il *Commesso* inserisce dati su alcuni campi della form e il sistema, effettuata la ricerca, completa i campi mancanti.

Control Object

Nome	Descrizione
RicercaCorsoControl	Gestisce l'interazione fra la form *RicercaCorsoForm* e gli entity object relativi.
RicercaLibroControl	Gestisce l'interazione fra la form *RicercaLibroForm* e gli entity object relativi.
RicercaFornitoreControl	Gestisce l'interazione fra la form *RicercaFornitoreForm* e gli entity object relativi.
RicercaLibriPrenotatiControl	Gestisce l'interazione fra la form *RicercaLibriPrenotatiForm* e gli entity object relativi.
RicercaPrenotazioniControl	Gestisce l'interazione fra la form *RicercaPrenotazioniForm* e gli entity object relativi.
RicercaLibroCorsoControl	Gestisce l'interazione fra la form *RicercaLibriDiCorsoForm* e gli entity object relativi.
RicercaOrdiniControl	Gestisce l'interazione fra la form *RicercaOrdiniForm* e gli entity object relativi.
AnalisiVenditeControl	Gestisce l'interazione fra la form *AnalisiVenditeForm* e gli entity object relativi.
DataBaseControl	Gestisce l'interazione fra l'attore *DataBase* e gli entity object relativi.
GestioneOrdineControl	Gestisce l'interazione fra la form *OrdineForm* e gli entity object relativi.
GestioneLibroControl	Gestisce l'interazione fra la form *RegistraLibroForm* e gli entity object relativi.
GestioneCorsoControl	Gestisce l'interazione fra la form *RegistraCorsoForm* e gli entity object relativi.
GestioneFornitoreControl	Gestisce l'interazione fra la form *RegistraFornitoreForm* e gli entity object relativi.
GestionePrenotazioneControl	Gestisce l'interazione fra la form *PrenotazioneForm* e gli entity object relativi.
GestioneVenditaControl	Gestisce l'interazione fra la form *VenditaForm* e gli entity object relativi.
GiacenzaMagazzinoControl	Gestisce l'interazione fra la form *GiacenzeMagazzinoForm* e gli entity object relativi.
HistoryControl	Gestisce l'interazione fra la form *HistoryForm* e gli entity object relativi.
StampaControl	Gestisce l'interazione fra l'attore *Stampante* e gli entity object relativi.

3.5.3.2. Class Diagrams

Si riporta il diagramma concettuale delle classi che mostra il dominio del sistema:

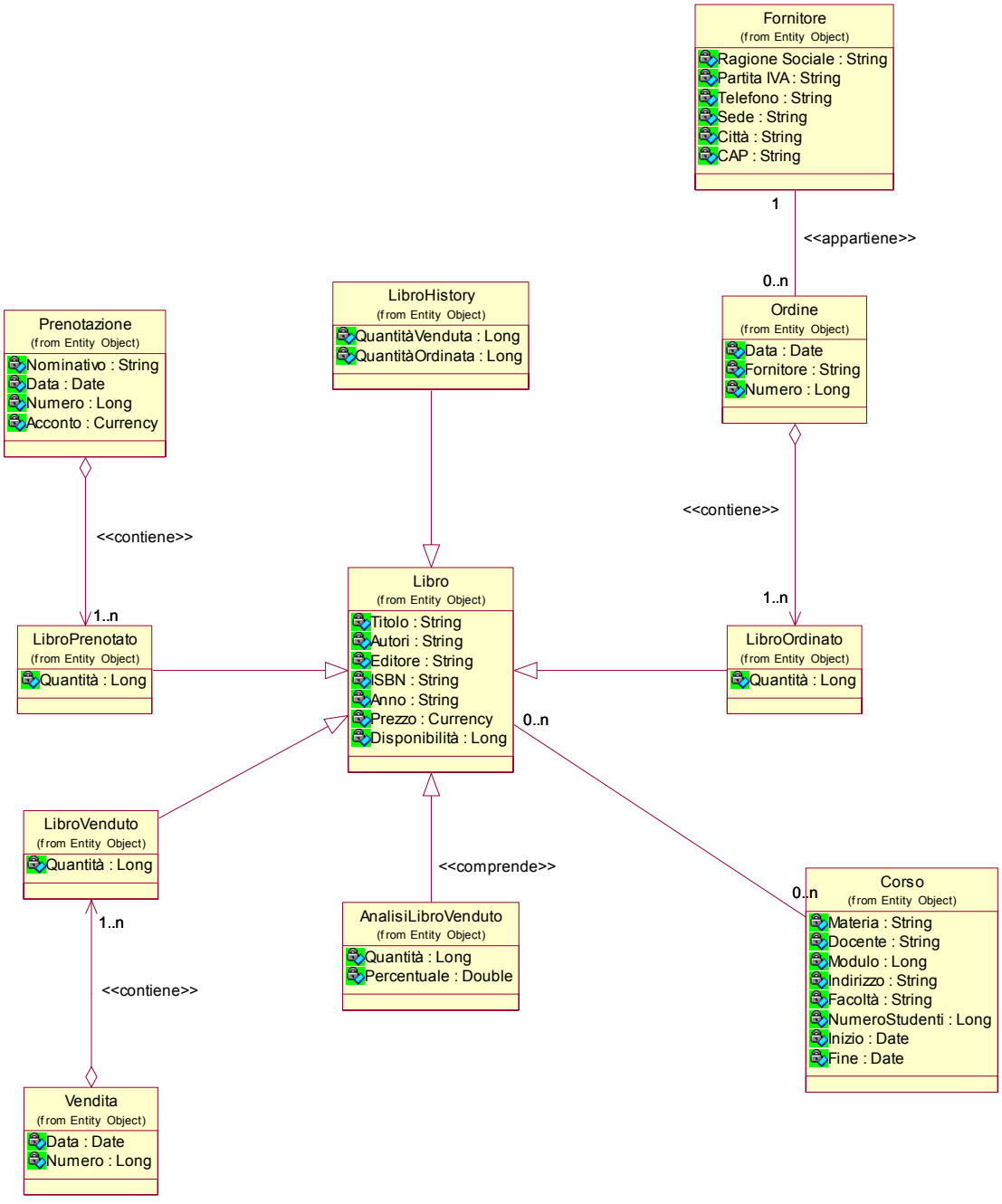


Figura 13 - Diagramma delle Classi (dominio dell'applicazione)

I seguenti class diagram mostrano le relazioni tra le classi di controllo del sistema nel dominio dell'applicazione:

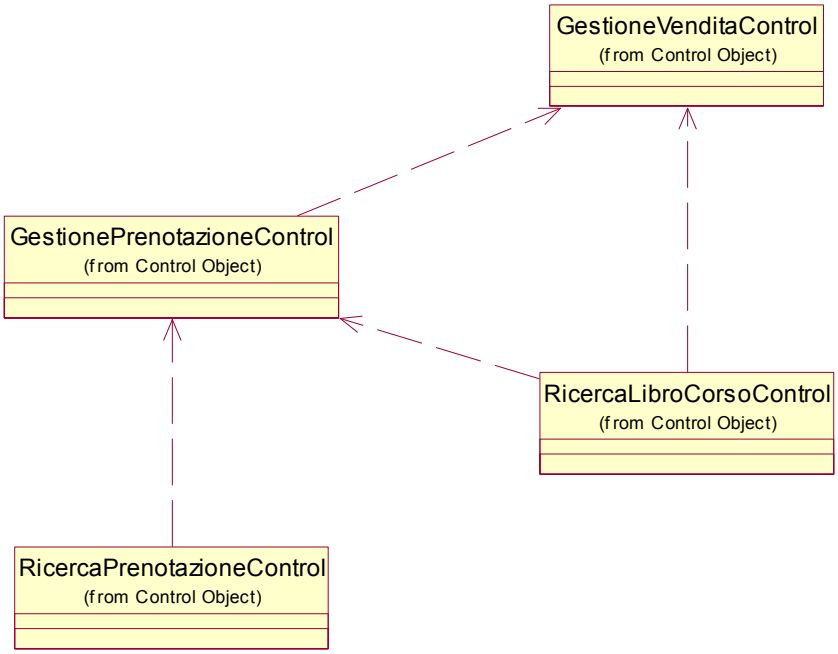


Figura 14 - Relazioni fra le classi di controllo per le operazioni tipiche del Commesso

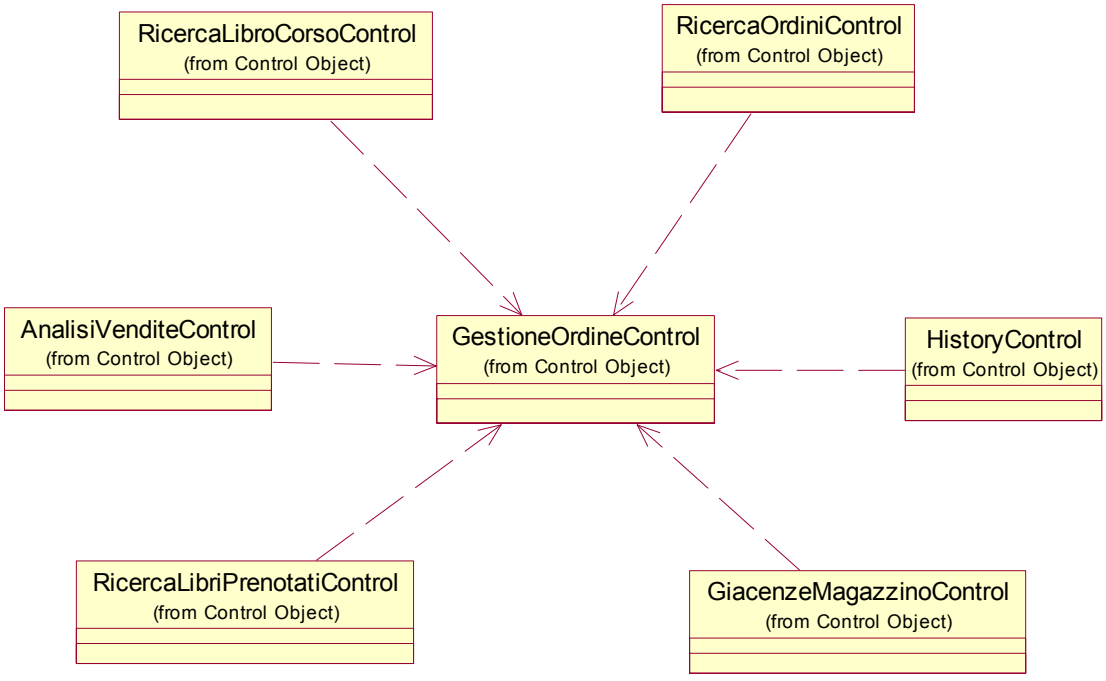


Figura 15 - Relazioni fra le classi di controllo per le operazioni tipiche del Responsabile

Figura 16 - Classi di controllo nella gestione dei corsi

Figura 17 - Classi di controllo nella gestione dei fornitori

Figura 18 - Classi di controllo nella gestione dei libri

Figura 19 - Classe di controllo per l'accesso al DataBase

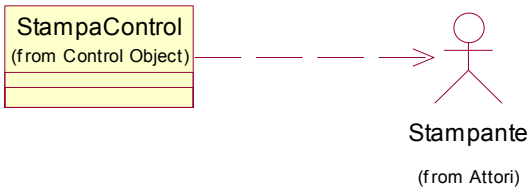


Figura 20 - Classe di controllo per l'accesso alla Stampante

3.5.4. Dynamic Models

I seguenti sequenze diagram descrivono la sequenza delle iterazioni tra i vari attori e gli oggetti del sistema:

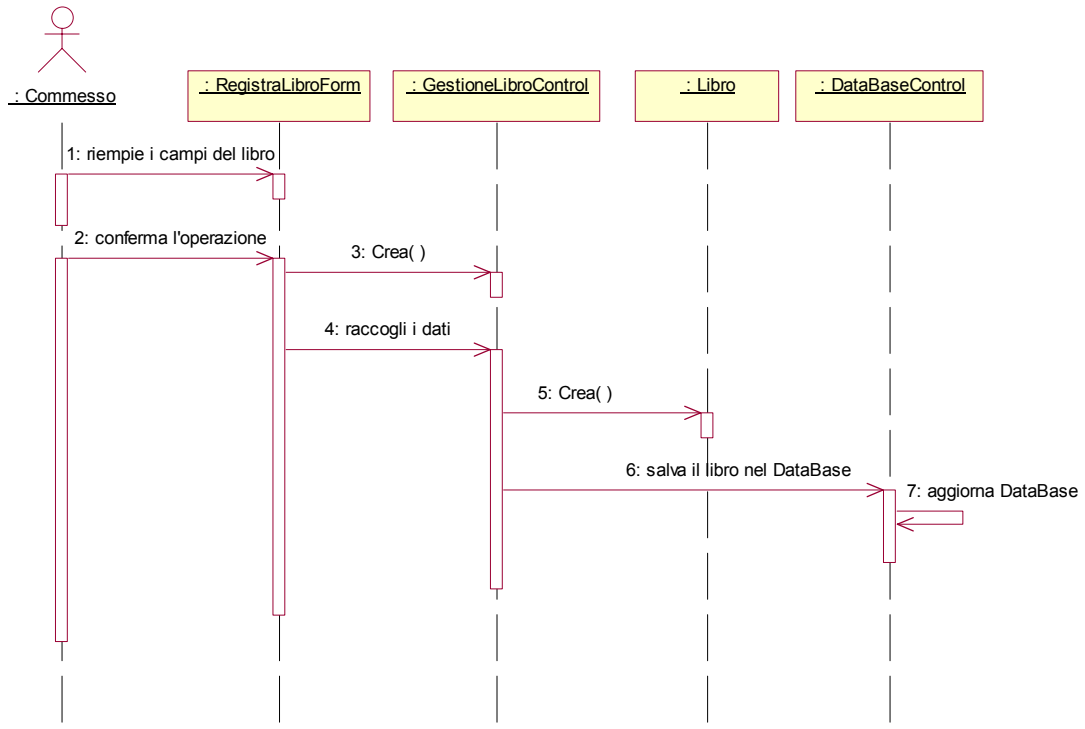


Figura 21 - Sequence diagram relativo alla registrazione di un nuovo libro

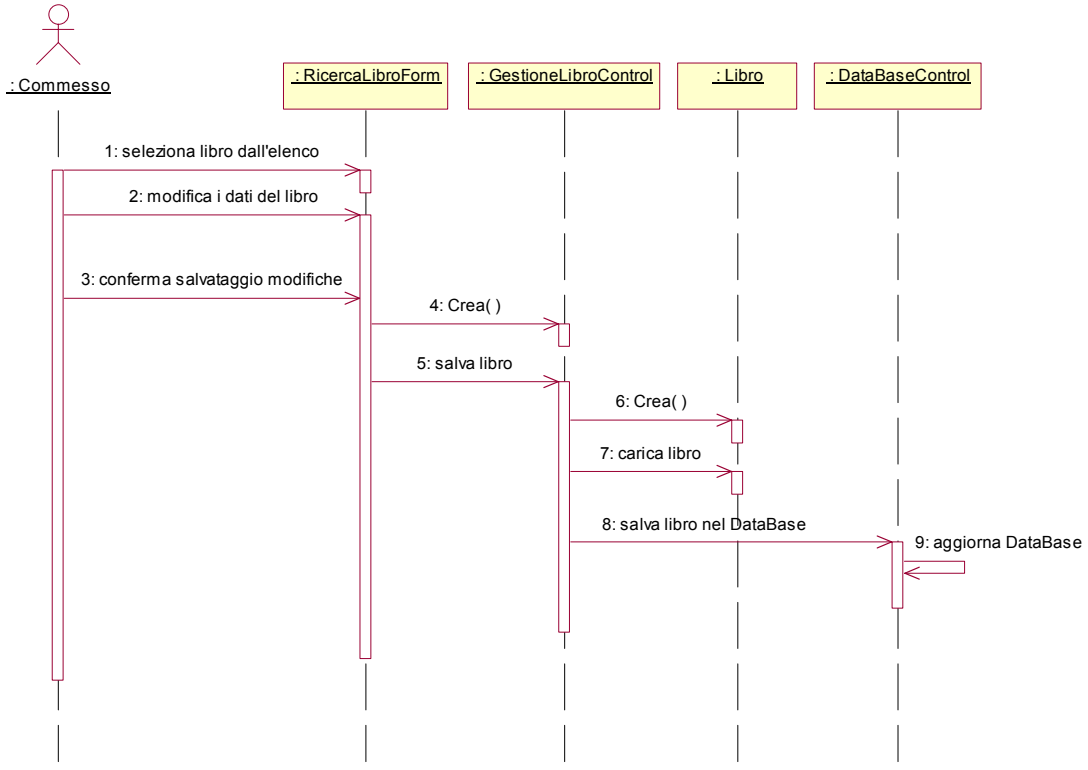


Figura 22 - Sequence diagram relativo alla modifica dei dati di un libro

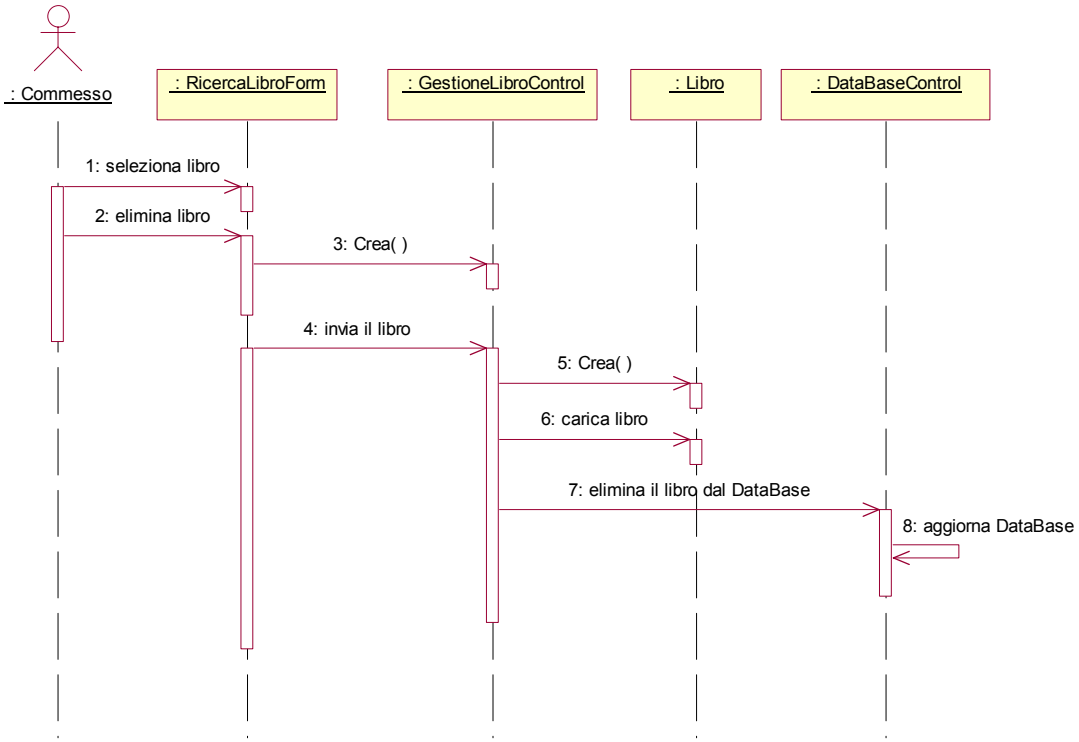


Figura 23 - Sequence diagram relativo all'eliminazione di un libro

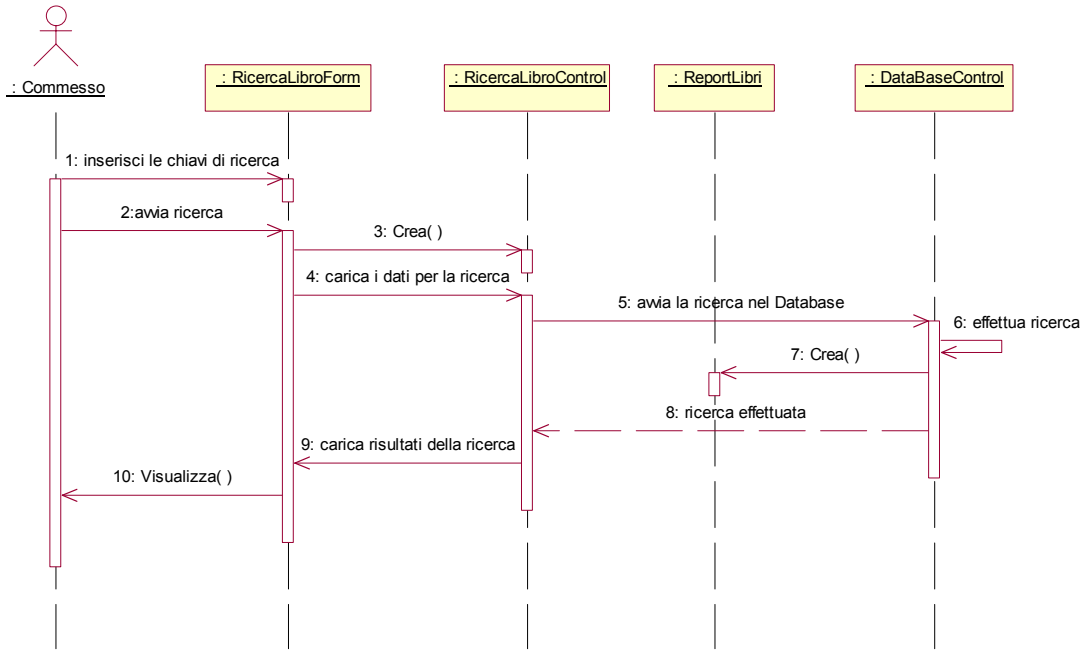


Figura 24 - Sequence diagram relativo alla ricerca di un libro

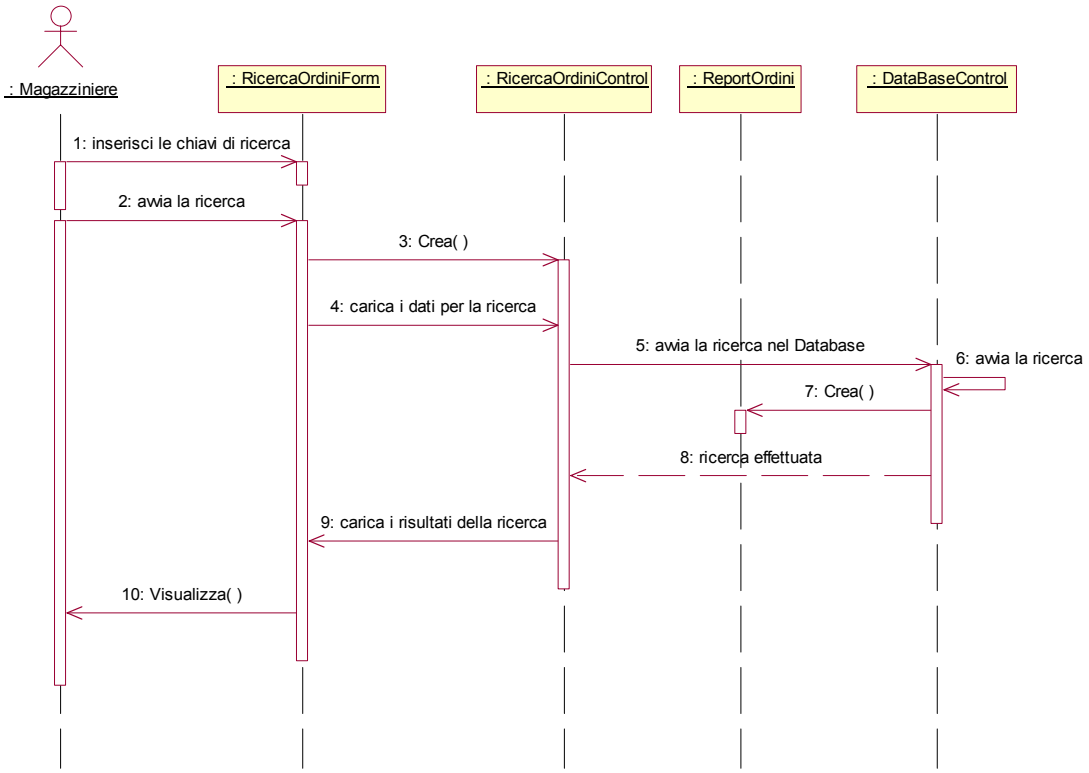


Figura 25 - Sequence diagram relativo alla ricerca di un ordine

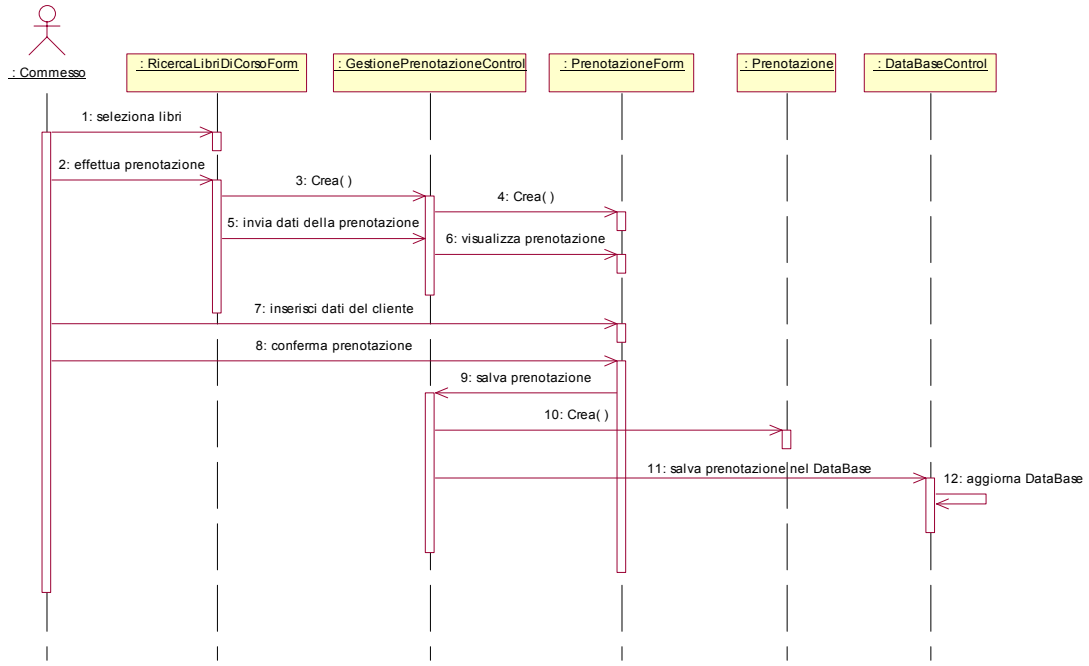


Figura 26 - Sequence diagram relativo ad una prenotazione

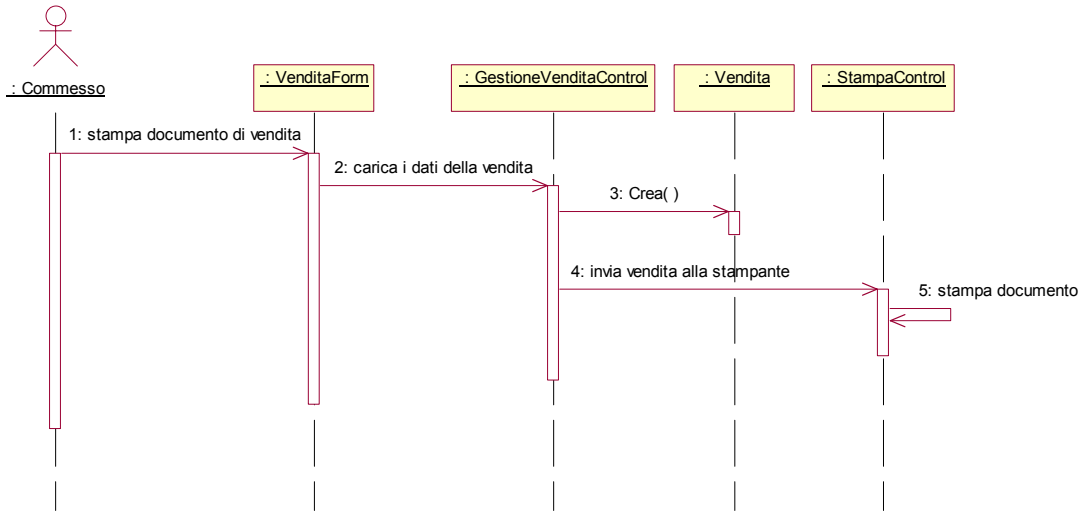


Figura 27 - Sequence diagram relativo alla stampa di un documento di vendita

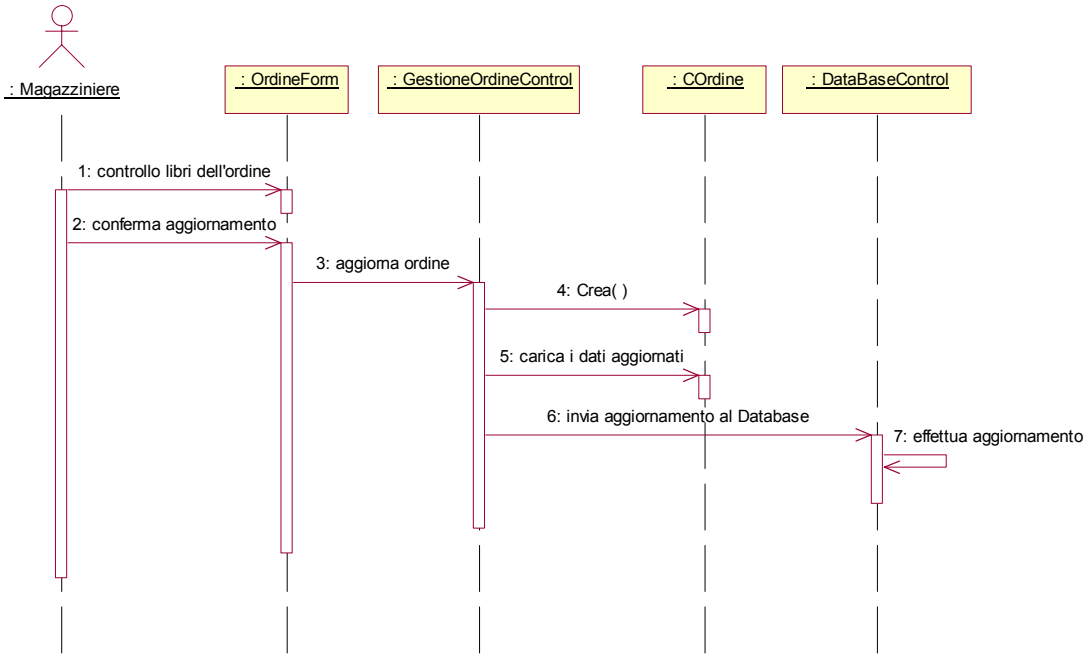


Figura 28 - Sequence diagram relativo alla modifica di un ordine

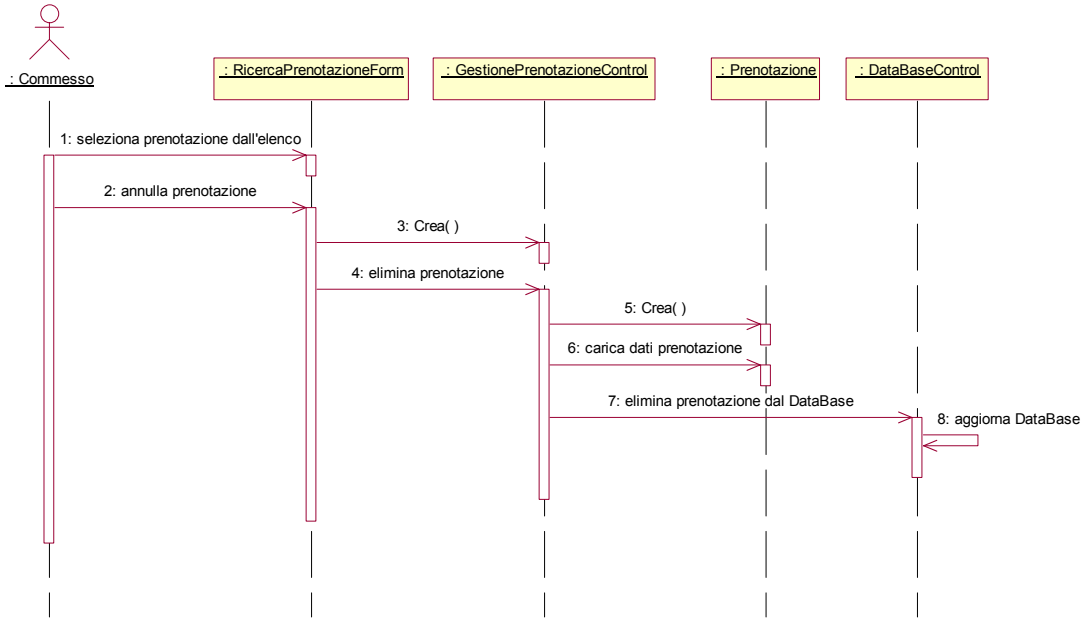


Figura 29 - Sequence diagram relativo all'eliminazione di una prenotazione

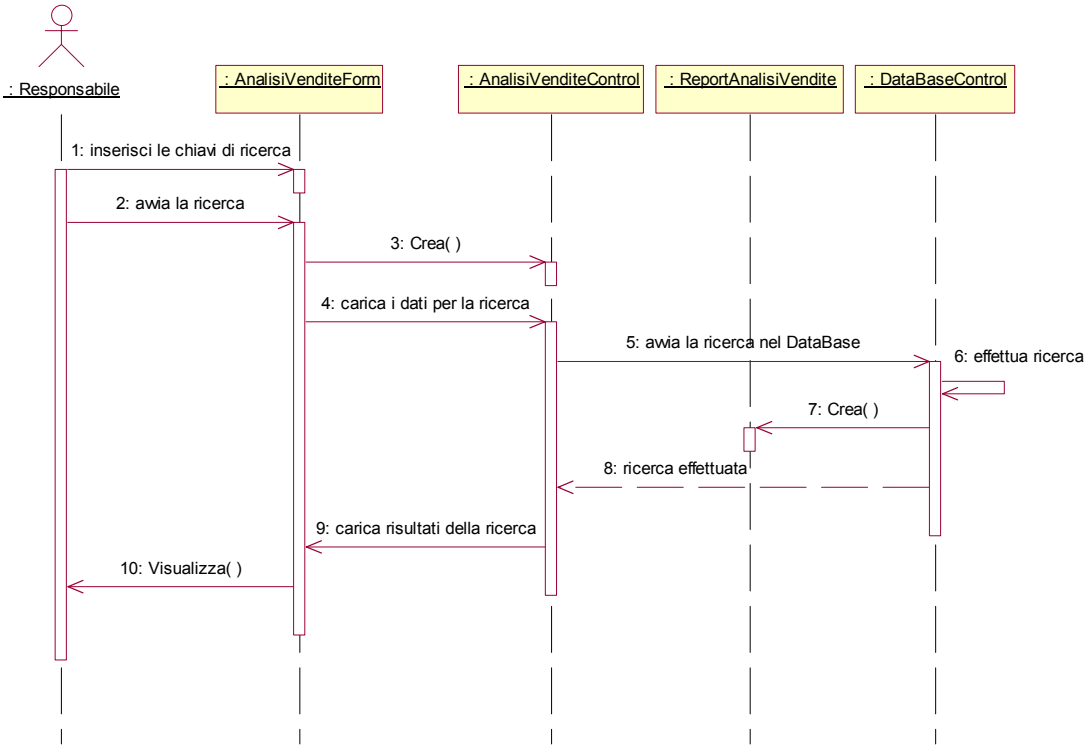


Figura 30 - Sequence diagram relativo all'analisi delle vendite dei libri

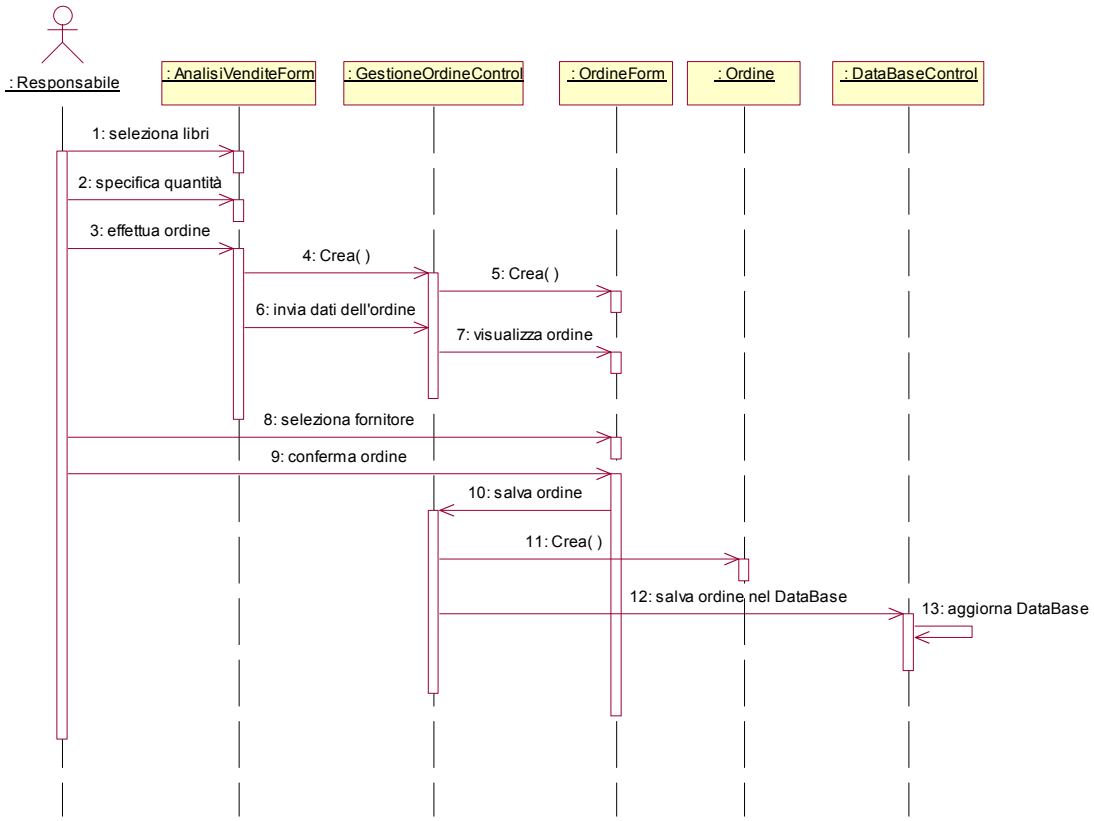


Figura 31 - Sequence diagram relativo al salvataggio di un ordine

I seguenti state diagram mostrano una vista generale delle funzionalità messe a disposizione dal sistema in relazione al tipo di utente:

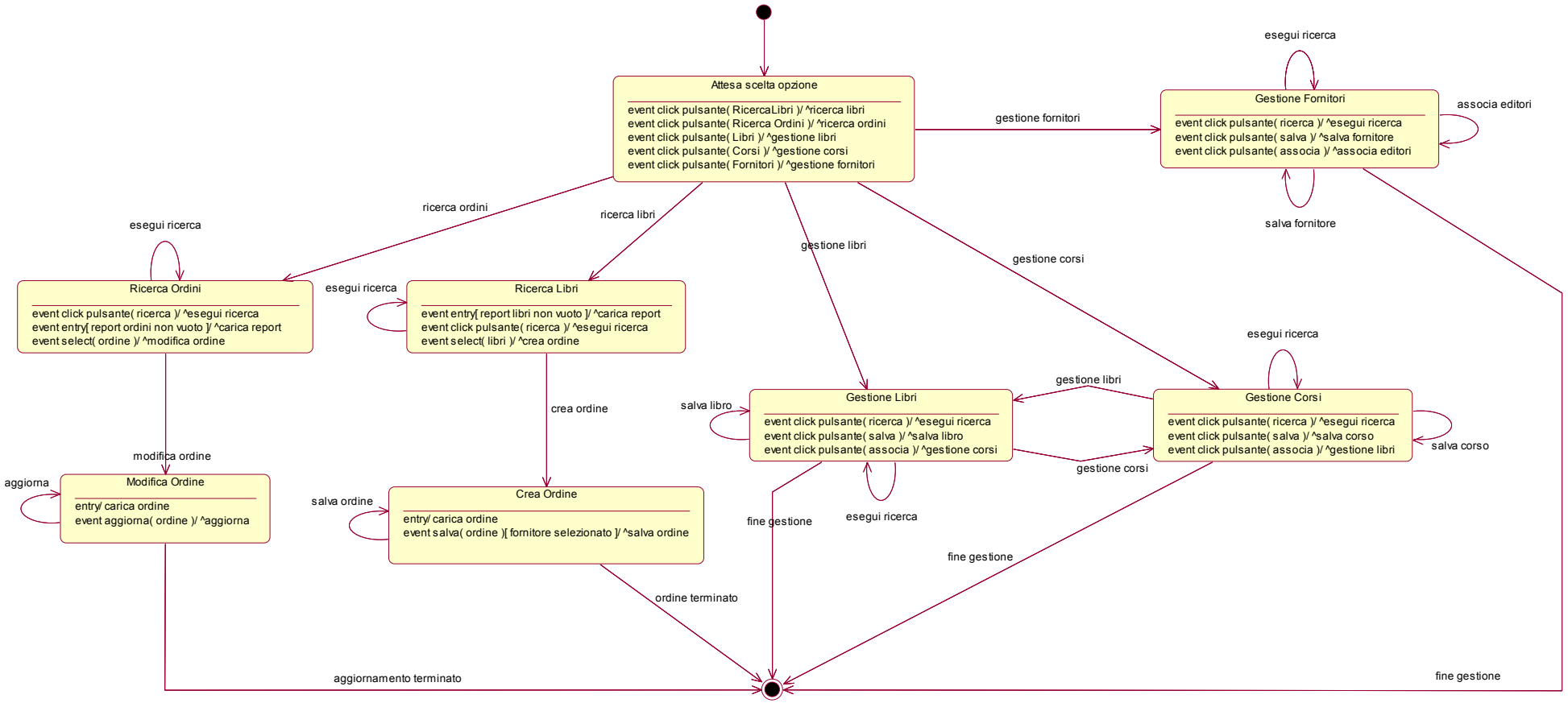


Figura 32 - State diagram relativo alle funzionalità del Responsabile

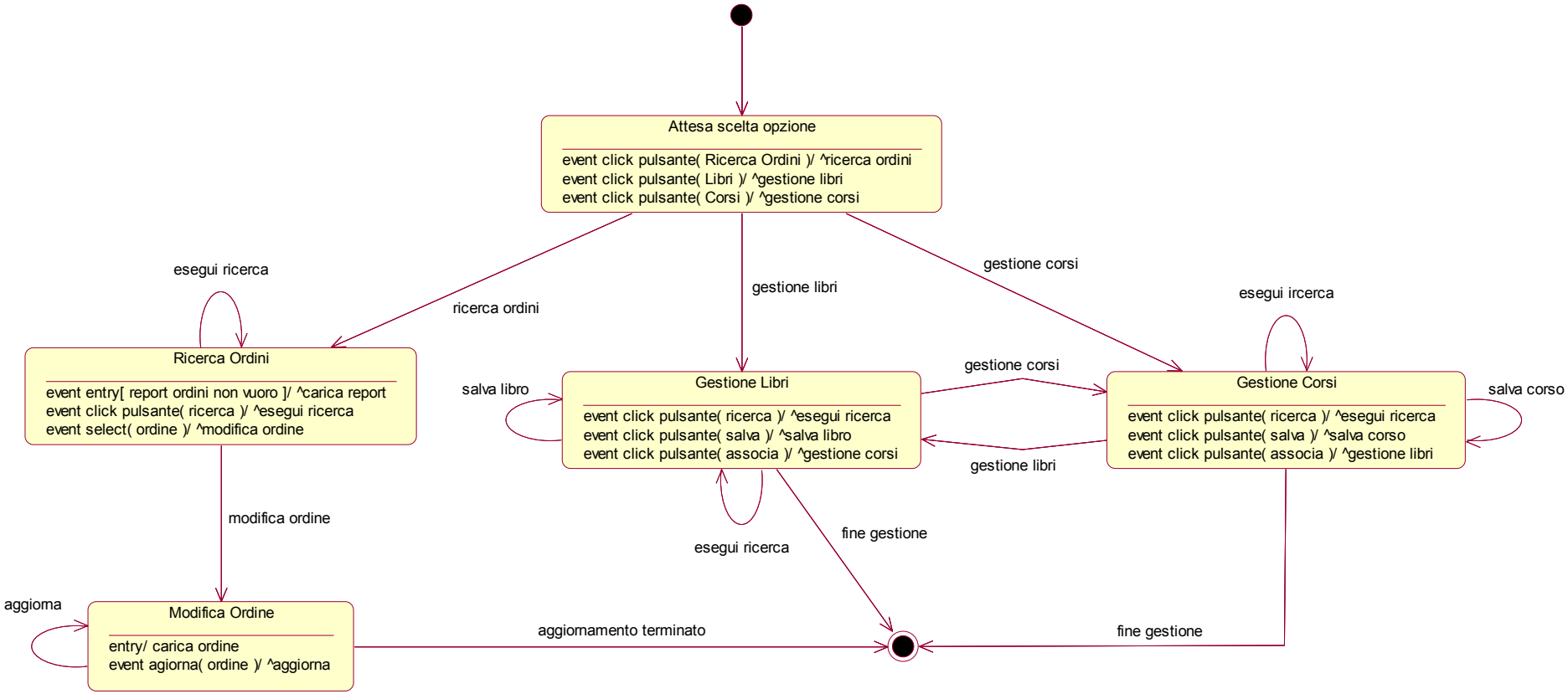


Figura 33 - State diagram relativo alle funzionalità del Magazziniere

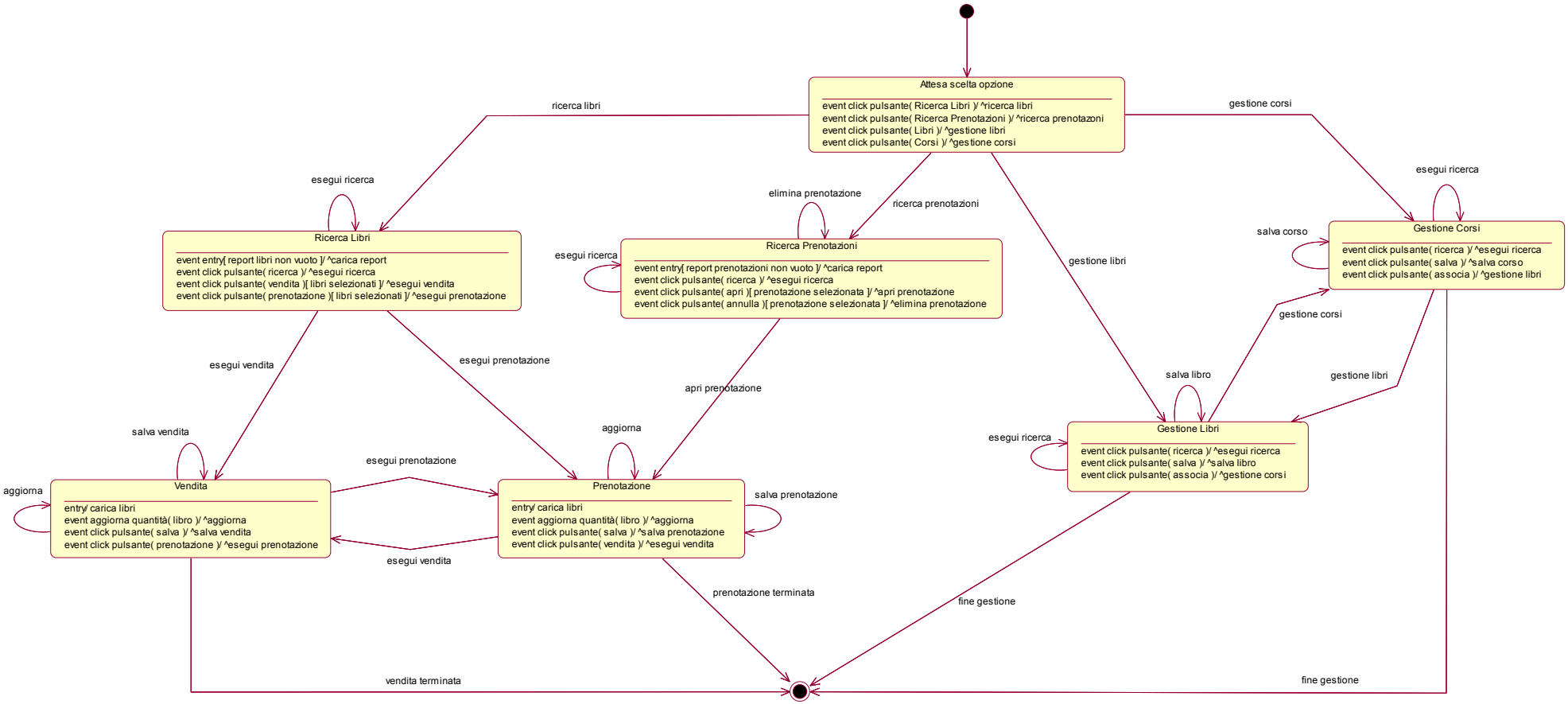


Figura 34 - State diagram relativo alle funzionalità del Commesso

3.5.5. Interfaccia Utente – Screen Mockups

Di seguito si riportano i mockups delle tre postazioni, Commesso, Magazziniere e Responsabile; tali mockups vogliono solo essere indicativi e di supporto ad una migliore comprensione ed eventuale revisione del software da parte del cliente, pertanto potranno essere soggetti a cambiamenti.

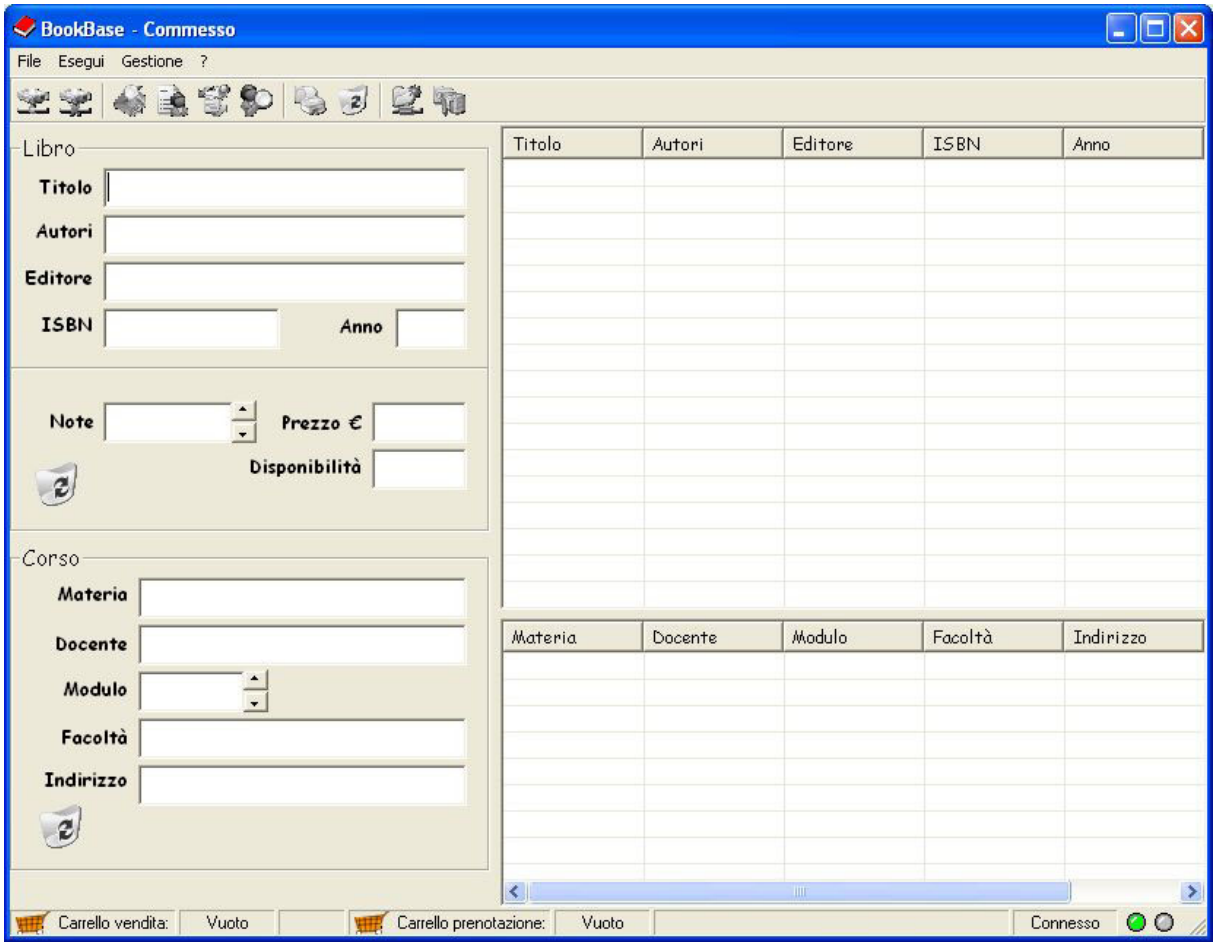


Figura 35 - Mockup della Postazione Commesso

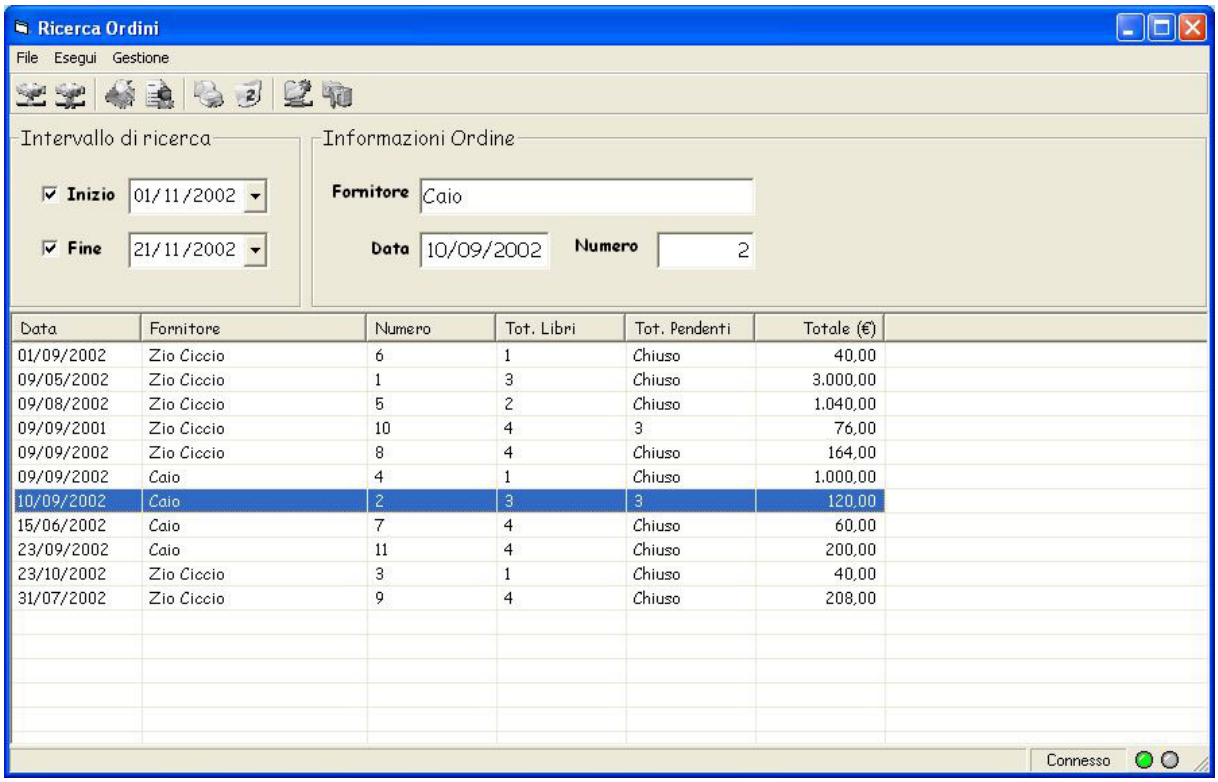


Figura 36 - Mockup della Postazione Magazzino

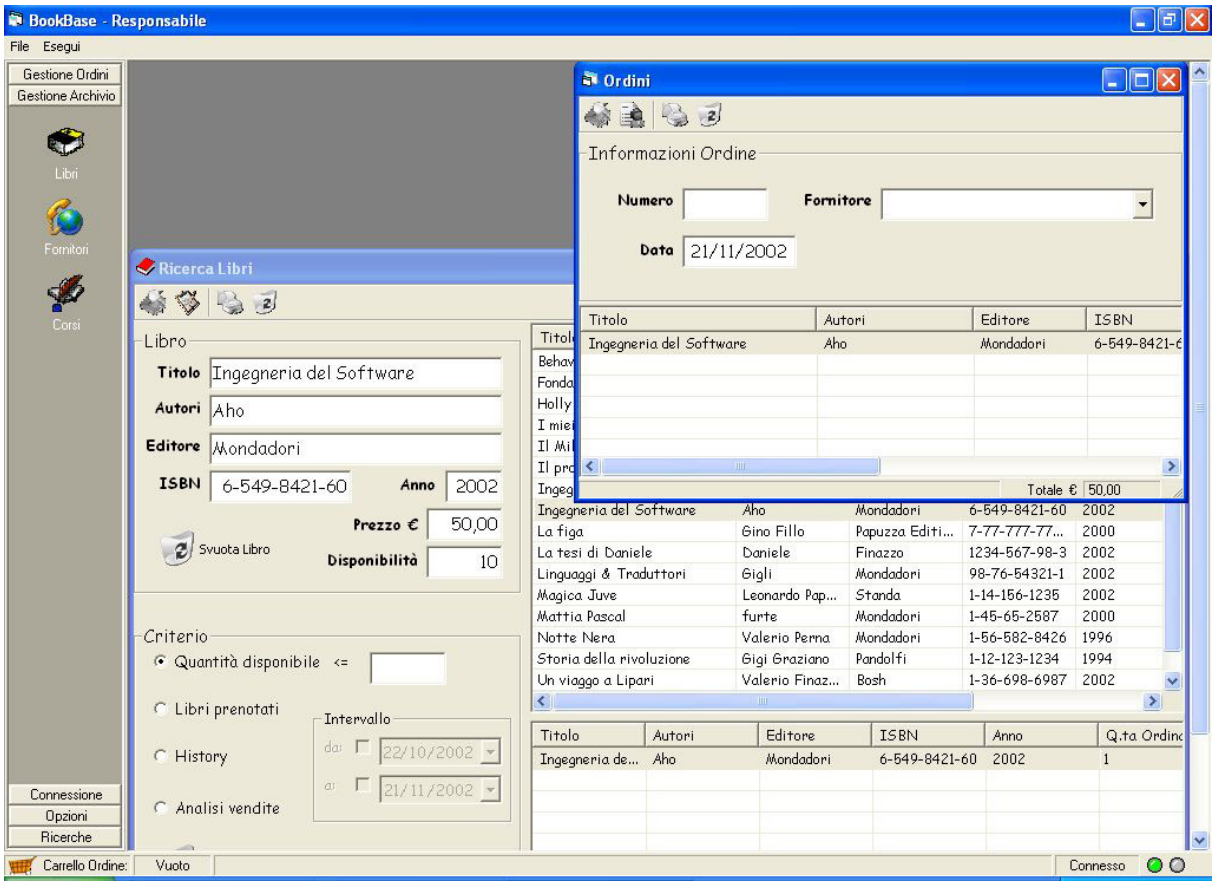


Figura 37 - Mockup della Postazione Responsabile

System Design Document

Revision History:

Versione R0.1, creazione: 13 Settembre 2002.
Versione R0.2, modifica: 12 Ottobre 2002.
Versione R1.0, modifica: 06 Dicembre 2002.

Preface:

Questo documento riporta l'analisi dell'architettura del software BOOKBASE; esso è rivolto agli analisti e ai programmatori.Esso descrive gli obietti progettuali, la decomposizione del sistema e il mapping hardware/software.

Target Audience:

Analisti, Programmatori

Project Members:

Valerio Finazzo, Leonardo Papuzza, Daniele Ribaudo

Indice

1. Obiettivi del Progetto

L'obiettivo del software è la realizzazione di un sistema in grado di rendere più agevole la gestione di una libreria universitaria. Il software deve poter essere eseguito in ambiente Windows e deve interagire con una base di dati contenente tutte le informazioni necessarie per il corretto svolgimento dell'attività della libreria. Per evitare impatti traumatici con gli utenti, il software deve presentare delle interfacce amichevoli, intuitive e di facile utilizzo.
Per ulteriori informazioni sulle funzionalità che il tool deve offrire si rimanda al documento di analisi dei requisiti (RAD).

2. Architettura Software Proposta

2.1. Overview

La scelta dell'architettura software è vincolata dall'analisi del sistema; si tratta di un sistema distribuito in cui è possibile distinguere dei moduli client, ognuno dei quali contiene un'interfaccia specializzata per ogni utente del sistema, e un modulo server, che offre un insieme di funzionalità necessarie per l'accesso e la modifica dei dati; naturalmente tali moduli risiedono in macchine differenti. La comunicazione tra client e server avviene utilizzando l'architettura Three-Tier (fig.1), evoluzione dell'architettura client/server. In questo tipo di approccio è contemplato uno strato intermedio (middleware) per la gestione del trasferimento dei dati dal livello client al livello archivio dati; tale modulo intermedio, nel nostro caso rappresentato dal sottosistema DataBase Management e dai driver ODBC[1], si occupa di inoltrare le richieste dell'utente al livello archivio dati (realizzato con MS-Access, di tipo MDB) e di trasmettere i relativi risultati. I vantaggi di tale architettura sono quelli di rendere indipendente il client dalle problematiche di accesso ai dati e di centralizzare tutto il software di gestione del database in un modulo appropriato. Questo aspetto fornisce all'intero sistema una facile e semplice manutenibilità e aggiornabilità del software per l'accesso al livello archivio dati. Resta da definire il tipo di protocollo utilizzato per la comunicazione client - middleware.

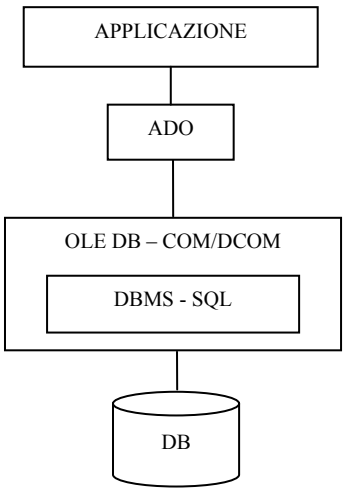


Figura 1 - Architettura Three - Tier

Avendo scelto Visual Basic come ambiente di sviluppo dell'intero progetto, si è optato per il protocollo standard ADO[2], invece che utilizzare le complesse e ostiche API ODBC. I vantaggi derivanti

[1] *Il noto Open DataBase Connectivity, motore di gestione per database relazionali in locale e remoto.*

[2] *ActiveX Data Objects, la nuova tecnologia basata su OLE che fornisce accesso a database relazionali tramite ODBC.*

dall'utilizzo di tale protocollo sono dovuti alla semplicità con cui avviene la chiamata agli svariati metodi e la standardizzazione a cui è stato sottoposto il protocollo stesso.

2.2. Decomposizione del Sistema

Il sistema BOOKBASE è composto da 5 sottosistemi. Di seguito si riporta il diagramma dei package del sistema e una breve descrizione di ciascun sottosistema.

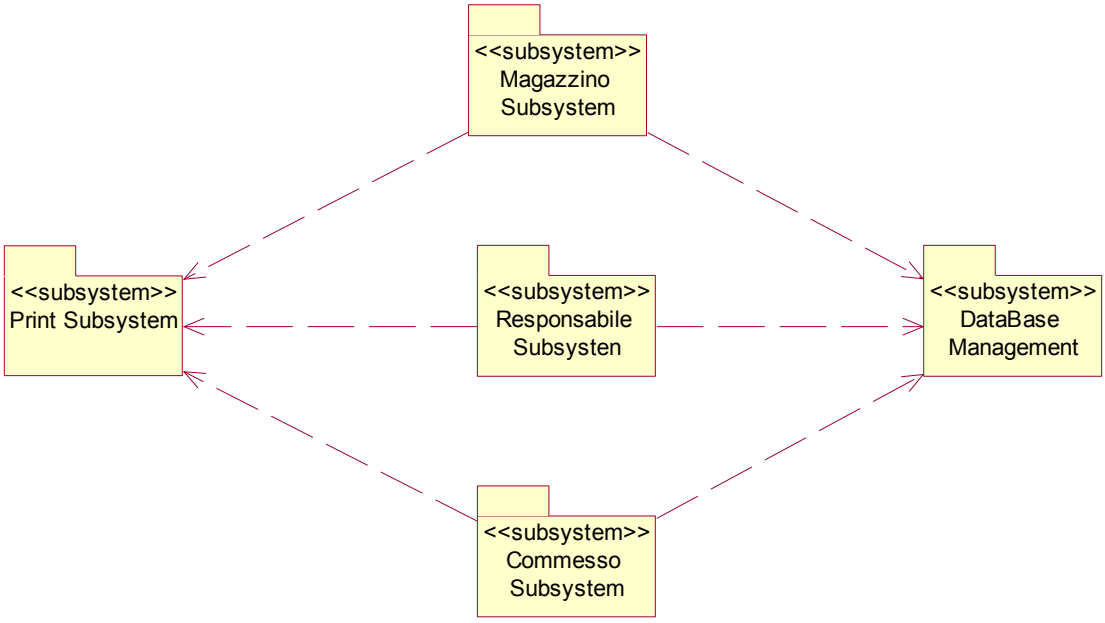


Figura 2 - Scomposizione in subsystem

2.2.1. Descrizione dei packages

2.2.1.1. Commesso Subsystem

Il *Commesso Subsystem* contiene classi che implementano l'insieme di operazioni delegate al supporto dell'attività del commercio di libri; in particolar modo tali classi permettono di realizzare le funzionalità riguardanti la consultazione del magazzino, la registrazione delle vendite, la richiesta di prenotazioni di libri.

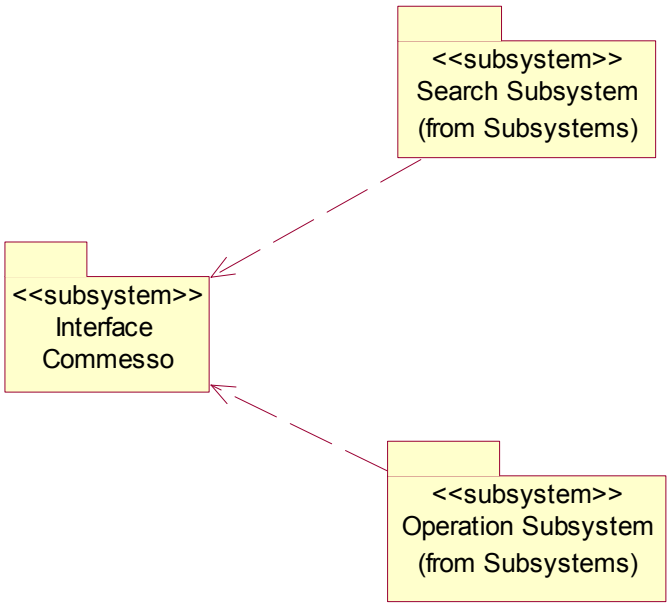


Figura 3 – Commesso SubSystem

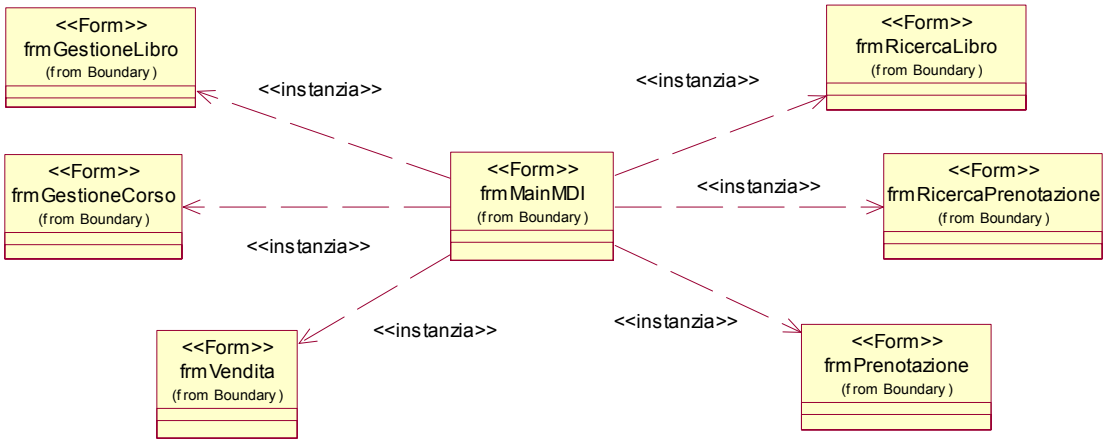


Figura 4 - Classi contenute nel sottosistema Interface Commesso

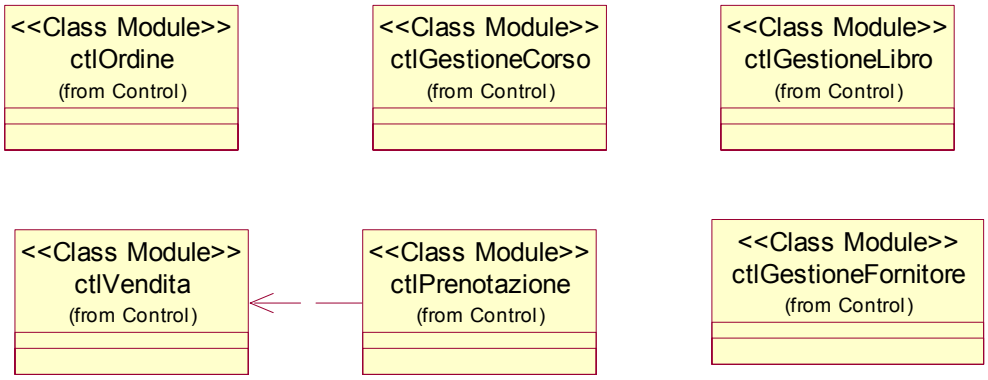


Figura 5 - Classi di controllo contenute nel sottosistema Operation Subsystem

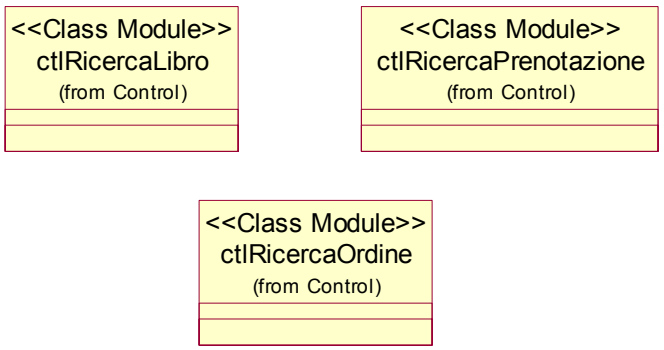


Figura 6 - Classi di controllo contenute nel sottosistema Search Subsystem

2.2.1.2. Magazzino Subsystem

Il *Magazzino Subsystem* contiene classi che implementano l'insieme di operazioni delegate al supporto dell'attività di gestione magazzino; in particolar modo tali classi permettono di realizzare le funzionalità riguardanti la registrazione delle consegne di libri da parte dei fornitori.

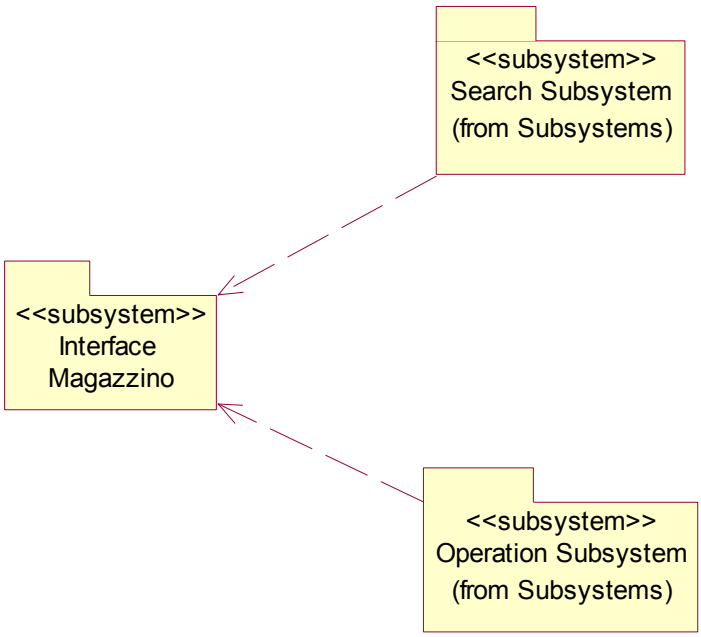


Figura 7 - Magazzino Subsystem

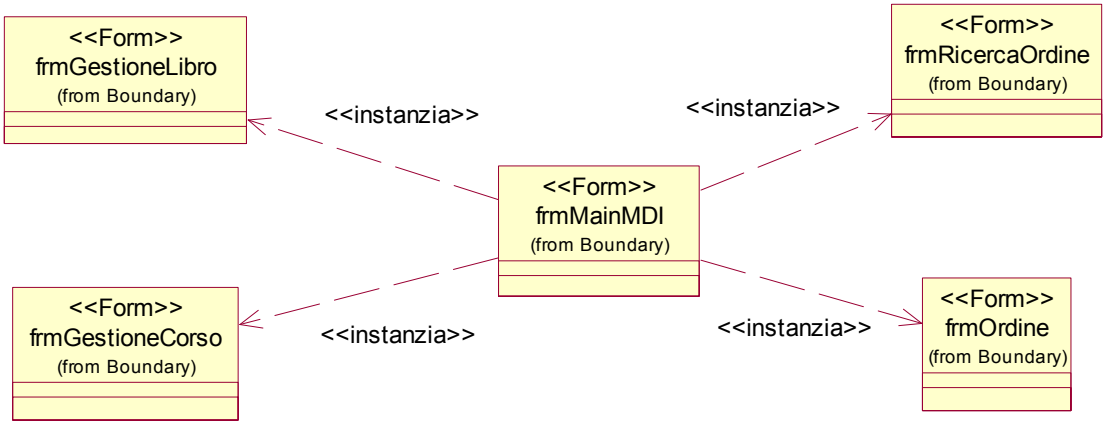


Figura 8 - Classi contenute nel sottosistema Interface Magazzino

2.2.1.3. Responsabile Subsystem

Il *Responsabile Subsystem* contiene classi che implementano l'insieme di operazioni delegate al supporto dell'attività del responsabile per un completo controllo dell'esercizio; in particolar modo tali classi permettono di realizzare le funzionalità riguardanti la consultazione del magazzino, l'effettuazione di ordinazioni di libri, la visualizzazione delle statistiche dell'esercizio (supporto decisionale).

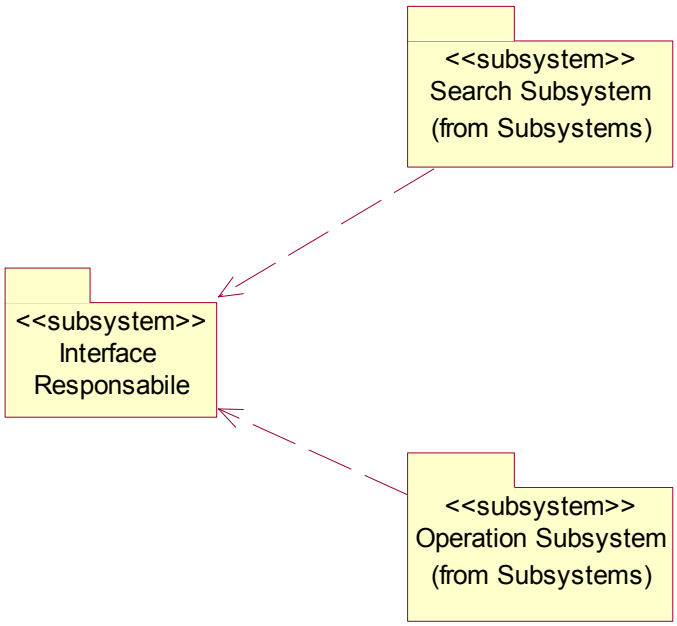


Figura 9 - Responsabile Subsystem

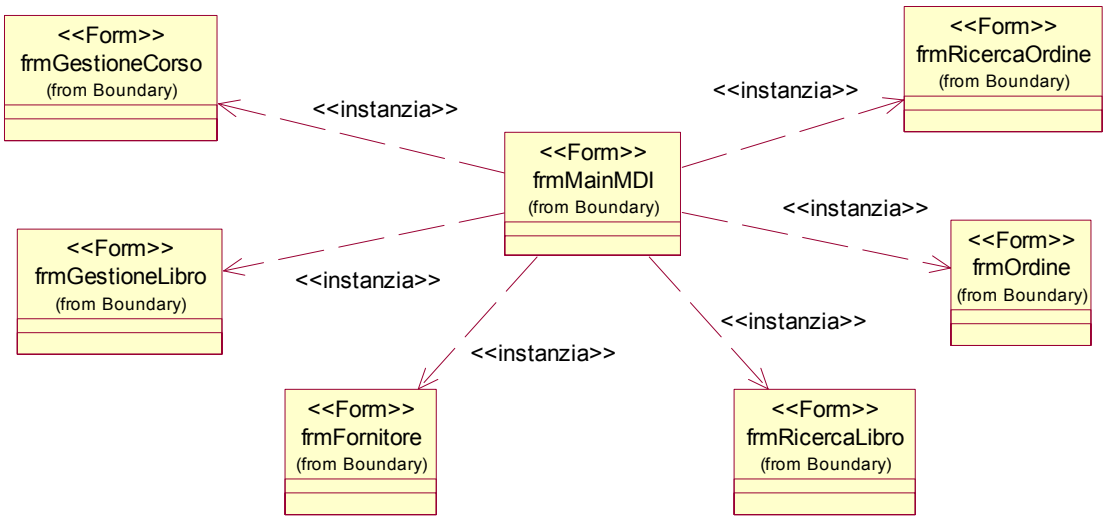


Figura 10 - Classi contenute nel sottosistema Interface Responsabile

2.2.1.4. DataBase Management

Il *DataBase Management* contiene classi che si occupano della comunicazione con il database della libreria; tutti gli altri sottosistemi fanno uso di tale sottosistema per inoltrare delle richieste ad "alto livello" di dati dal database. Il sottosistema si preoccupa di tradurre tali richieste in linguaggio SQL e di fornire i dati estratti in una forma tale da poter essere elaborati dai sottosistemi richiedenti. Inoltre il sottosistema controlla per ogni transazione da e verso il DB lo stato di connessione attuale, informando l'utente se la connessione con il DB è andata persa.

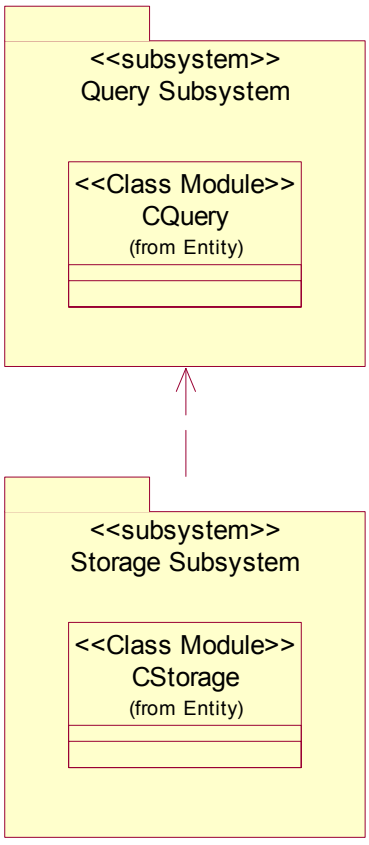


Figura 11 - DataBase Management

2.2.2. Topologia del sistema

Illustriamo la topologia di BOOKBASE nel seguente deployment view:

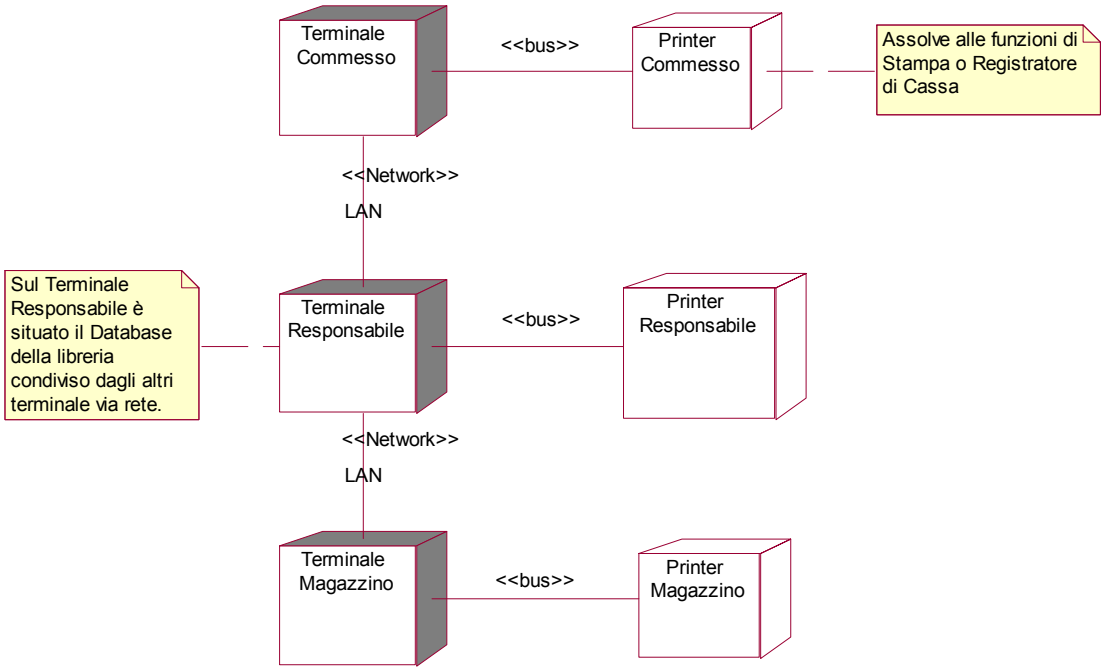


Figura 12 - Deployment view del sistema

3. Hardware/Software Mapping

I terminali necessari al sistema proposto non presentano particolari richieste hardware per assolvere alle funzionalità implementate nel software.
Di seguito si suggerisce, senza voler imporre determinati vincoli di scelta, dei modelli di componenti hardware ritenuti adatti alle necessità.

- N° 3 Terminali (Personal Computer Assemblati):
  - Processore: AMD o INTEL based (>1 Ghz);
  - Ram: 256 Mb;
  - HDD: 40 Gb per le postazioni vendite e magazzino, 80 Gb per la postazione ordini;
  - Scheda Video: Agp con almeno 32 Mb di memoria;
  - CD-RW 24x10x40;
  - Scheda Ethernet 100 MBit/s;
  - Modem/Fax V.90, 4 USB, 1 Parallela, 1 Seriale;
  - Monitor 15" LCD;
- N° 3 Stampanti Laser, una connessa al terminale di gestione e l'altra connessa al terminale vendite;
- N° 1 Hub Fast Ethernet;
- Cavo UTP, plug-in RJ45 e terminatori sufficienti al cablaggio dell'intera rete.

Non essendo state effettuate richieste esplicite di requisiti non funzionali da parte del committente, visto che il sistema informatico non presenta particolari esigenze ne in termini di un eventuale funzionamento multipiattaforma, ne in termini di eventuali richieste di capacità real-time, le uniche prestazioni richieste per la macchina dell'utente sono quelle necessarie per installare il software.
In particolar modo BOOKBASE sarà composto da 3 singole componeti software ognuna indipendente e ognuna in esecuzione in una postazione diversa. Il software in esecuzione nel Terminale Commesso implementerà il *Commesso Subsystem* e *DataBase Management*, quello che sarà in esecuzione nel Terminale Magazzino implementerà il *Magazzino Subsystem* e *DataBase Management*, infine il software in esecuzione nel Terminale Responsabile implementerà il *Responsabile Subsystem* e *DataBase Management*.

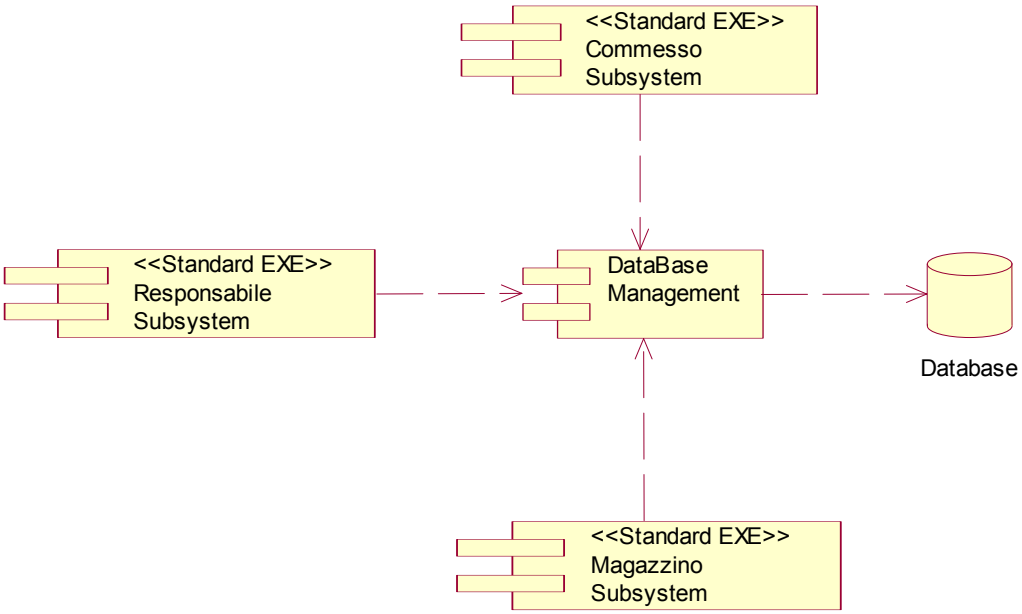


Figura 13 - Component diagram del sisyema

4. Gestione Dati

E' stata realizzata una base di dati condivisa, il cui accesso è gestito tramite un DBMS (Data Base Management System) sfruttato dal software BOOKBASE.
Il progetto prevede un archivio contenente tutti i dati di interesse per l'esercizio del committente, dai dati che caratterizzano i libri ai dati fondamentali per un buon supporto alle decisioni.
La progettazione della struttura delle tabelle costituenti il database è stata condotta con l'ausilio di Microsoft Access®, mentre il diagramma relazionale (ERD), riportato in fig.8, è stato realizzato con Microsoft Visio®.
Per ulteriori dettagli riguardanti il database, si rimanda alla documentazione relativa alla progettazione e realizzazione del database riportata in appendice.

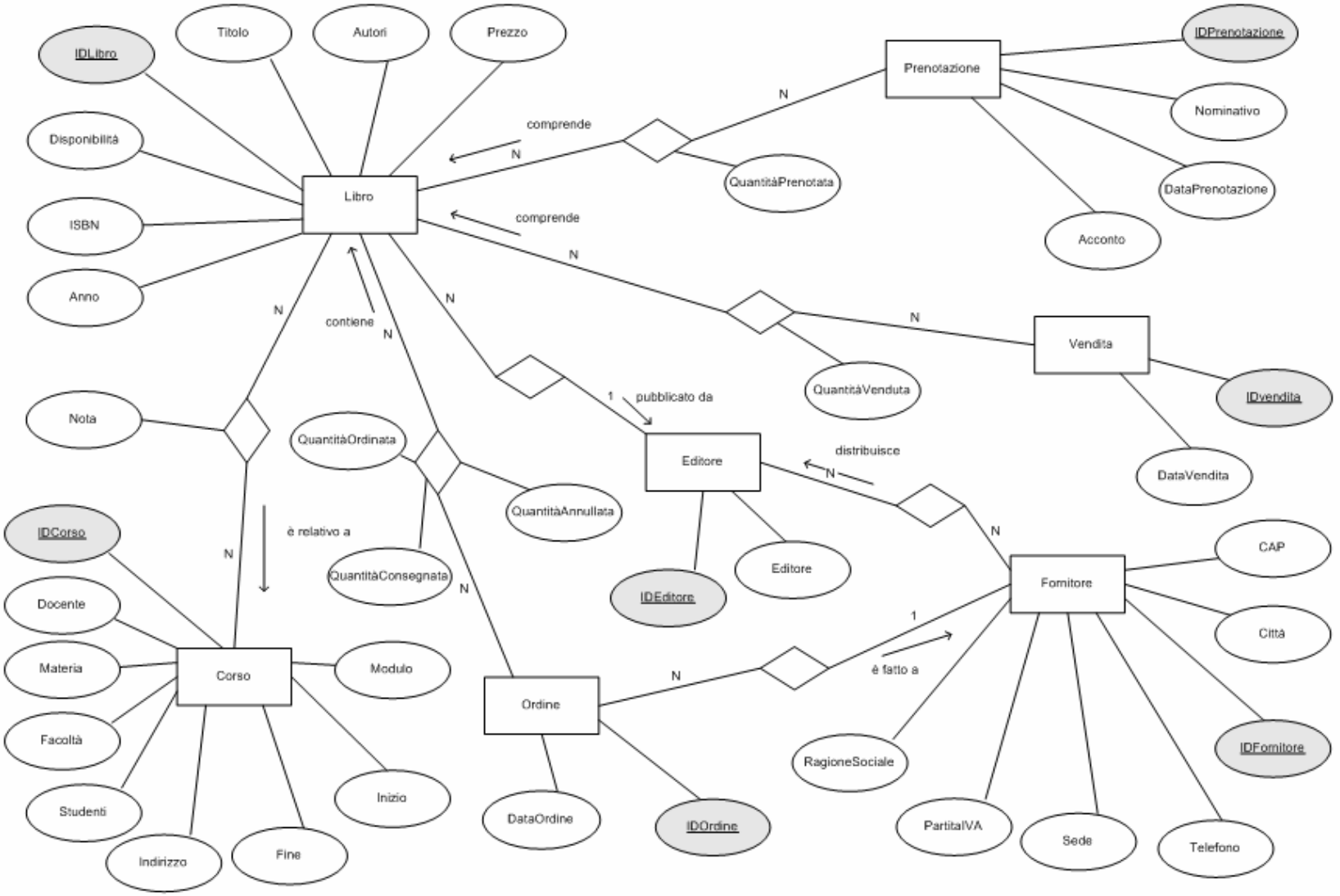


Figura 14 - ERD

Object Design Document

Revision History:

Versione R0.1, creazione: 26 Settembre 2002.
Versione R0.2, modifica: 22 Ottobre 2002.
Versione R0.3, modifica: 25 Ottobre 2002.
Versione R1.0, modifica: 06 Dicembre 2002.

Preface:

Questo documento indica i requisiti e le funzionalità del progetto BOOKBASE; esso è rivolto agli sviluppatori e ai clienti del progetto.

Target Audience:

Sviluppatori, Cliente

Project Members:

Valerio Finazzo, Leonardo Papuzza, Daniele Ribaudo

Indice

1. Diagramma delle Classi

Di seguito si riportano i class diagram relativi ai vari sottosistemi individuati nel System Design Document; nella sezione Class Interfaces si procederà all'analisi di ogni singola classe.

1.1. Commesso Subsystem

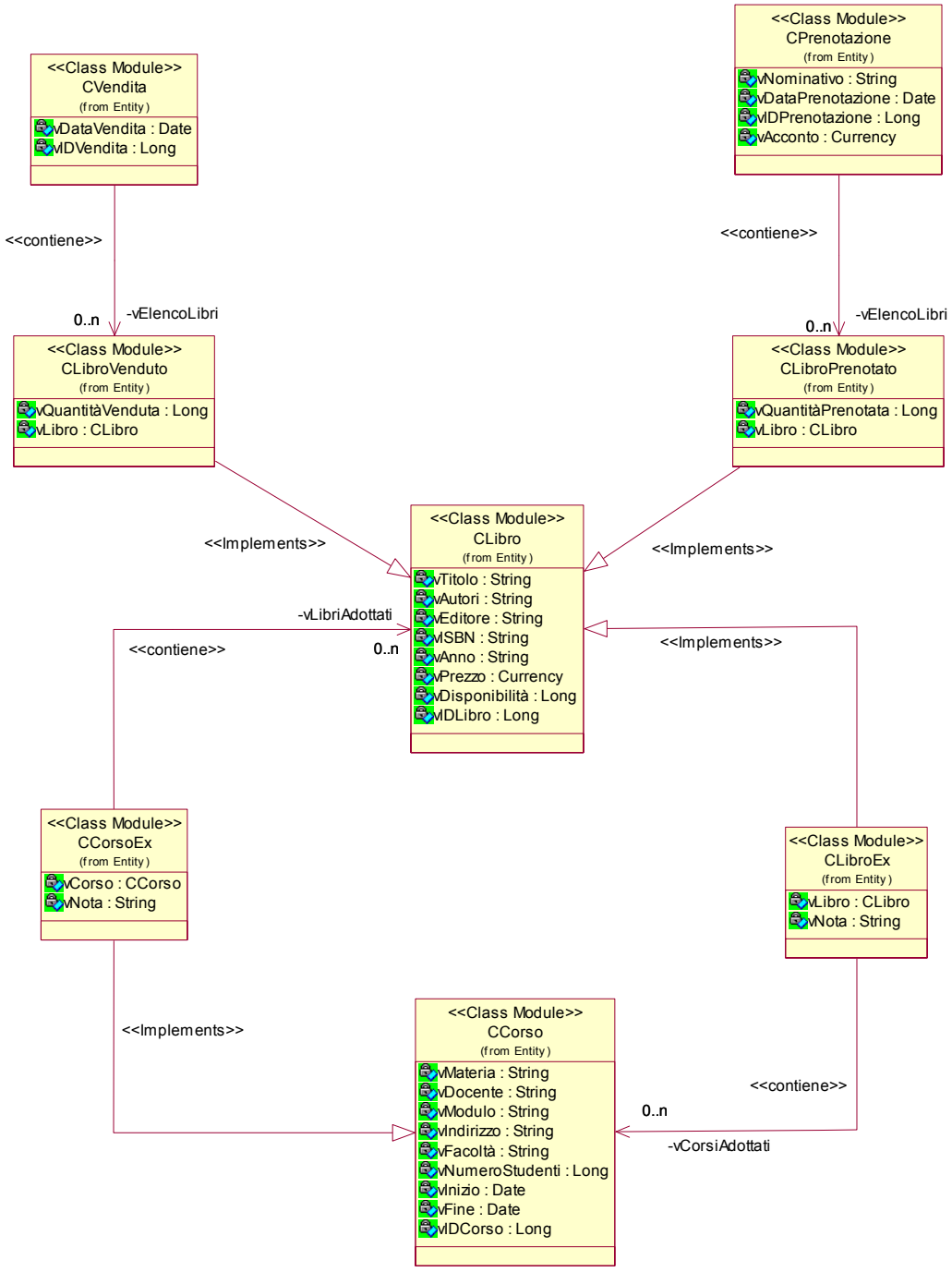


Figura 1 - Class diagram relativo al Commesso Subsystem

1.2. Magazzino Subsystem

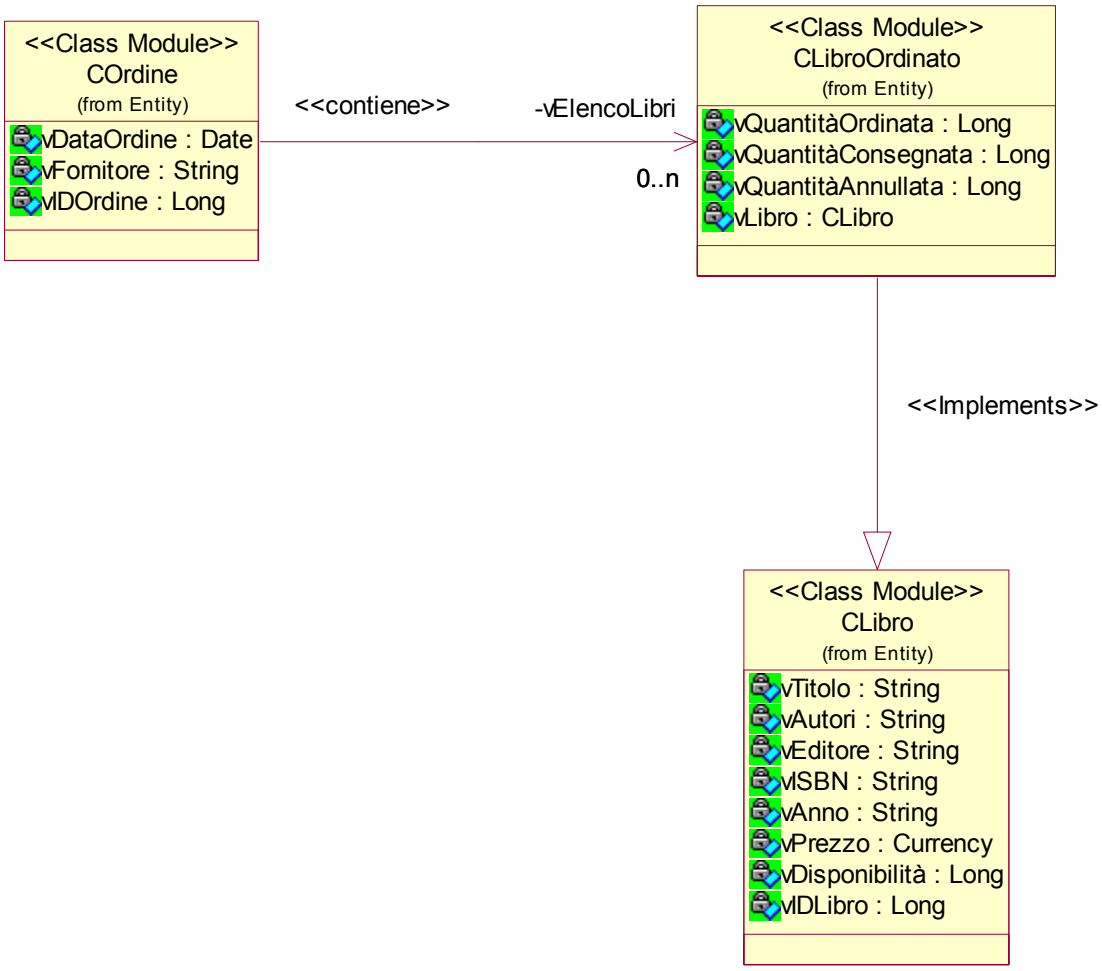


Figura 2 - Class diagram relativo al Magazzino Subsystem

1.3. Responsabile Subsystem

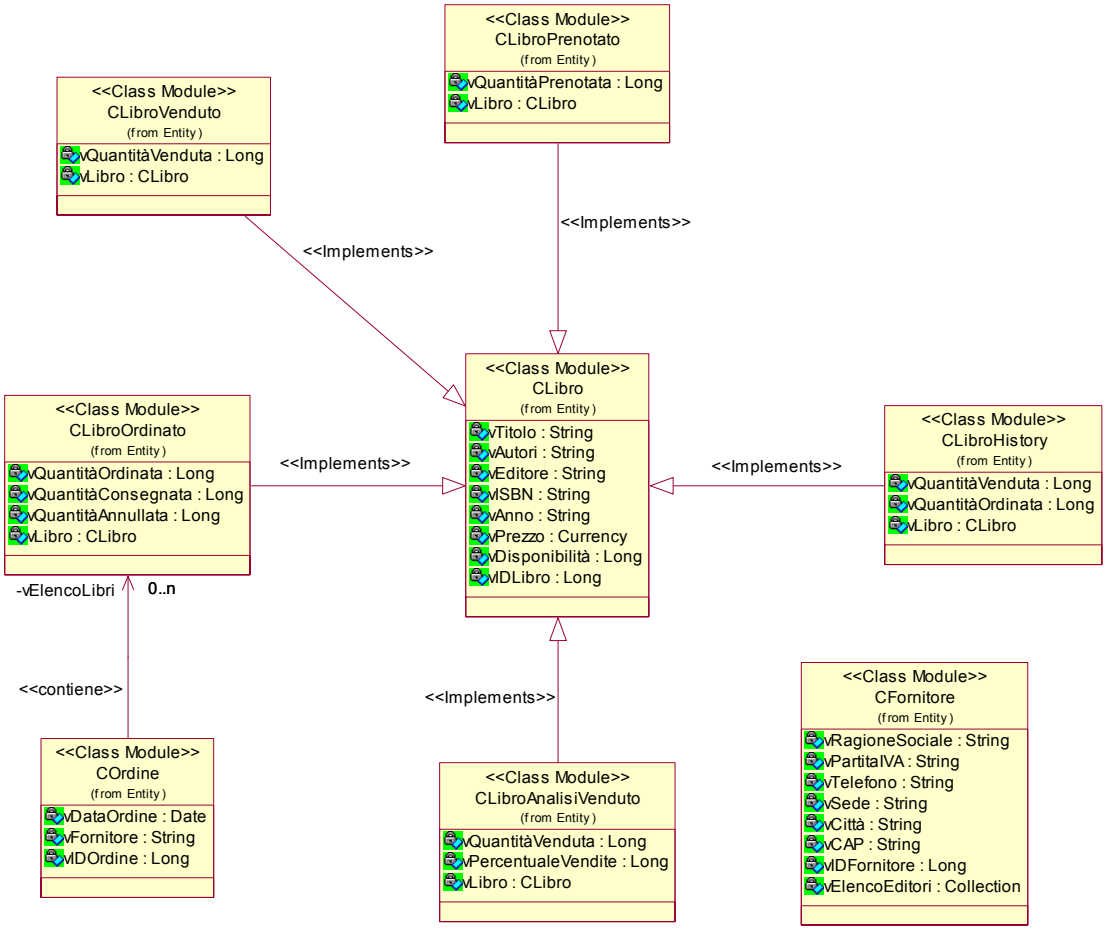


Figura 3 - Class diagram relativo al Responsabile Subsystem

1.4. DataBase Management

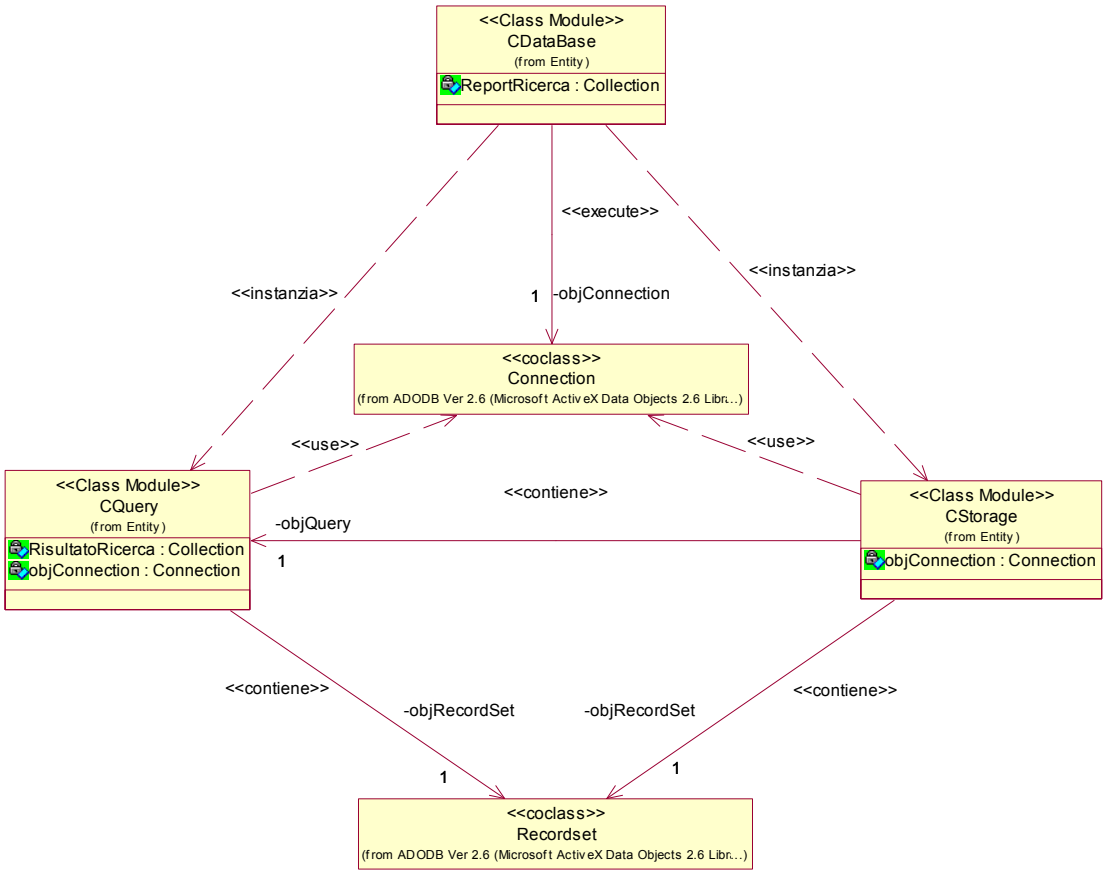


Figura 4 - Class diagram relativo al DataBase Management

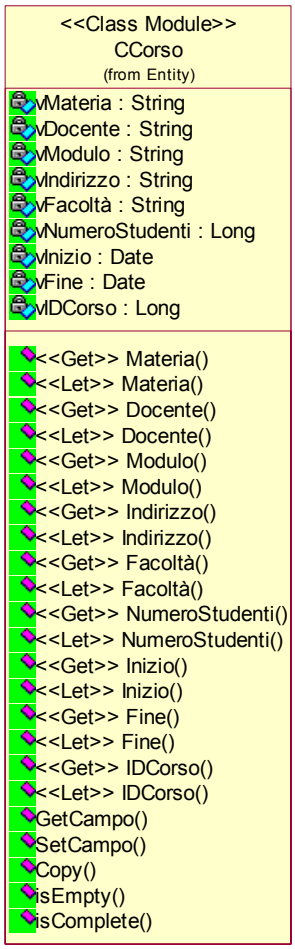
2. Class Interfaces

Si procede all'analisi dettagliata delle singole classi implementate nel sistema; l'analisi verte ad evidenziare le interfacce di interazione utilizzate nella progettazione del software.

2.1. CCorso

La classe contiene le informazioni relative ad un generico corso.

Attributi



- *Private vMateria As String*
  Il nome della materia.

- *Private vDocente As String*
  Il nome del docente titolare della materia.

- *Private vModulo As String*
  Il numero del modulo della materia (Primo, Secondo, Unico).

- *Private vIndirizzo As String*
  L'indirizzo di studi del corso in questione.

- *Private vFacoltà As String*
  La facoltà di appartenenza del corso in questione.

- *Private vNumeroStudenti As Long*
  Il numero di studenti del corso.

- *Private vInizio As Date*
  La data di inizio corso.

- *Private vFine As Date*
  La data di fine corso.

- *Private vIDCorso As Long*
  L'ID del corso; consente di identificare in maniera univoca il corso.

Metodi

- *Public Function GetCampo(Index As String) As Variant*
  In base all'indice *Index* passato come argomento, restituisce il valore dell'attributo corrispondente. Il campo Index può assumere i seguenti valori:
  ```
  Materia
  Docente
  Modulo
  Indirizzo
  Facoltà
  NumeroStudenti
  Inizio
  Fine
  IDCorso
  ```

- *Public Function SetCampo(Index As String, value As Variant) As Boolean*
  Assegna il valore *value* al campo identificato dall'indice *Index* restituendo TRUE se l'operazione è andata a buon fine, FALSE altrimenti.

- *Public Function Copy() As CCorso*
  Consente di ottenere il riferimento ad una copia dell'oggetto CCorso.

- *Public Function IsEmpty() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CCorso sono vuoti.

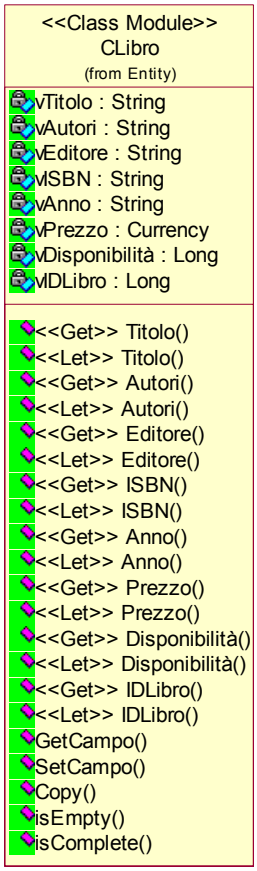
- *Public Function IsComplete() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CCorso sono stati riempiti.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.

2.2. CLibro

La classe contiene le informazioni relative ad un generico libro.

Attributi



- *Private vLibro As String*
  Il titolo del libro.
- *Private vAutori As String*
  L'autore del libro.
- *Private vEditori As String*
  L'editore del libro.
- *Private vISBN As String*
  L'ISBN del libro.
- *Private vAnno As String*
  L'anno di pubblicazione del libro.
- *Private vPrezzo As Currency*
  Il prezzo del libro.
- *Private vDisponibilità As Long*
  La disponibilità in magazzino del libro.
- *Private vIDLibro As Long*
  L'ID del libro; consente di identificare in maniera univoca il libro.

Metodi

- *Public Function GetCampo(Index As String) As Variant*
  In base all'indice *Index* passato come argomento, restituisce il valore dell'attributo corrispondente. Il campo Index può assumere i seguenti valori:

```
Titolo
Autori
Editore
ISBN
Anno
Prezzo
Disponibilità
IDLibro
```

- *Public Function SetCampo(Index As String, value As Variant) As Boolean*
  Assegna il valore *value* al campo identificato dall'indice *Index* restituendo TRUE se l'operazione è andata a buon fine, FALSE altrimenti.

- *Public Function Copy() As CLibro*
  Consente di ottenere il riferimento ad una copia dell'oggetto CLibro.

- *Public Function IsEmpty() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CLibro sono vuoti.

- *Public Function IsComplete() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CLibro sono stati riempiti.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.

2.3. CLibroEx

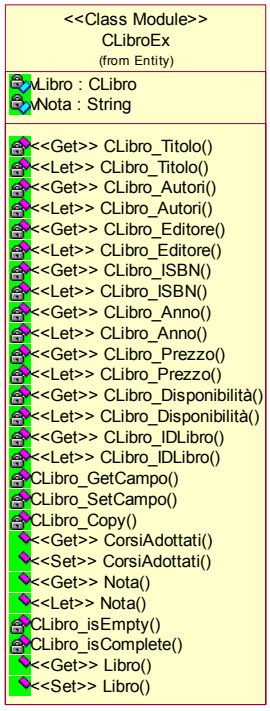
La classe contiene le informazioni relative ad un libro e ai corsi che lo adottano; in questi il libro può essere consigliato o richiesto. Tale classe eredita dalla classe CLibro attributi e metodi.

Attributi

- *Private vNota As String*
  Specifica se nei corsi che adottano il libro, questo è consigliato o richiesto.

- *Private vCorsiAdottati As Collection*
  La lista contente i corsi che adottano il libro in questione.

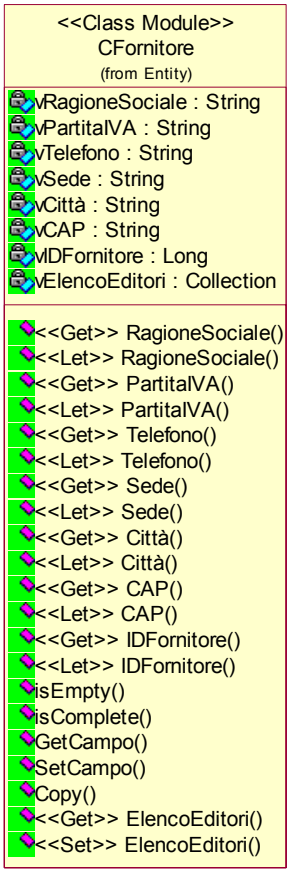
Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.



2.4. CFornitore

La classe contiene le informazioni relative ad un generico fornitore.

Attributi



- *Private vRagioneSociale As String*
  Il nome del fornitore.

- *Private vPartitaIVA As String*
  La partita IVA del fornitore.

- *Private vTelefono As String*
  Il numero di telefono del fornitore.

- *Private vSede As String*
  L'indirizzo della sede di distribuzione del fornitore.

- *Private vCittà As String*
  La città in cui ha sede il fornitore.

- *Private vIDFornitore As Long*
  L'ID del fornitore; consente di identificare in maniera univoca il fornitore.

- *Private vElencoEditori As Collection*
  Lista contenente gli editori distribuiti dal fornitore.

Metodi

- *Public Function GetCampo(Index As String) As Variant*
  In base all'indice *Index* passato come argomento, restituisce il valore dell'attributo corrispondente. Il campo Index può assumere i seguenti valori:
  `RagioneSociale`
  `PartitaIVA`
  `Telefono`
  `Sede`
  `Città`
  `CAP`
  `IDFornitore`

- *Public Function SetCampo(Index As String, value As Variant) As Boolean*
  Assegna il valore *value* al campo identificato dall'indice *Index* restituendo TRUE se l'operazione è andata a buon fine, FALSE altrimenti.

- *Public Function Copy() As CFornitore*
  Consente di ottenere il riferimento ad una copia dell'oggetto CFornitore.

- *Public Function IsEmpty() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CFornitore sono vuoti.

- *Public Function IsComplete() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CFornitore sono stati riempiti.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.

2.5. CLibroAnalisiVenduto

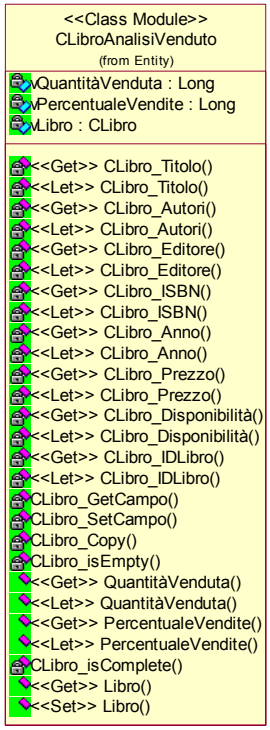
La classe contiene le informazioni relative ad un libro e ai dati statistici di supporto al responsabile per l'analisi delle vendite; essa eredita dalla classe CLibro attributi e vendite.

Attributi

- *Private vQuantitàVenduta As Long*
  Indica l'ammontare della quantità venduta per il libro in questione.

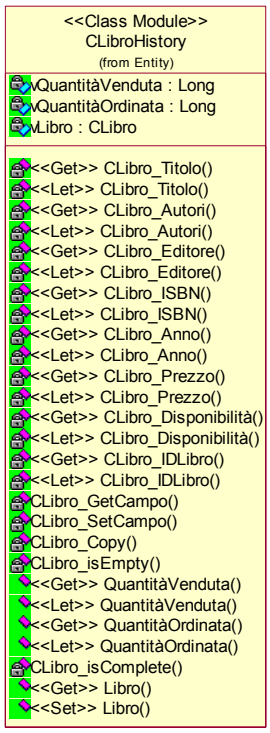
- *Private vPercentualeVendite As Long*
  Rapporto Q.tà_venduta/giacenze_magazzino in percentuale in un dato periodo di riferimento per il libro in questione.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.



2.6. CLibroHistory

La classe contiene le informazioni relative ad un libro e ai dati statistici di supporto al responsabile per l'history; essa eredita dalla classe CLibro attributi e metodi.



Attributi

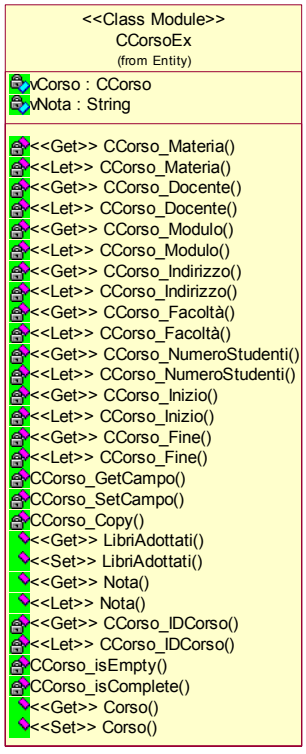
- *Private vQuantitàVenduta As Long*
  Indica l'ammontare della quantità venduta per il libro in questione, in riferimento ad un dato periodo.

- *Private vQuantitàOrdinata As Long*
  Indica l'ammontare della quantità ordinata per il libro in questione, in riferimento ad un dato periodo.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.

2.7. CCorsoEx

La classe contiene le informazioni relative ad un corso e ai libri da esso adottati, sia essi consigliati o richiesti. Tale classe eredita dalla classe CCorso attributi e metodi.



Attributi

- *Private vNota As String*
  Specifica se i libri adottati sono consigliati o richiesti nel corso.
- *Private vLibriAdottati As Collection*
  La lista contente i libri adottati nel corso in questione.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.

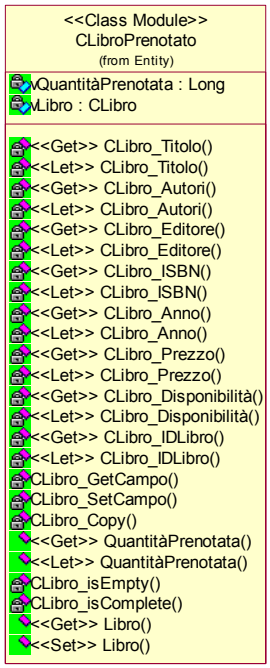
2.8. CLibroPrenotato

La classe contiene le informazioni relative ad un libro presente in una determinata prenotazione; essa eredita dalla classe CLibro metodi e attributi.

Attributi

- *Private vQuantitàPrenotata As Long*
  Indica l'ammontare della quantità prenotata per il libro in questione.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.



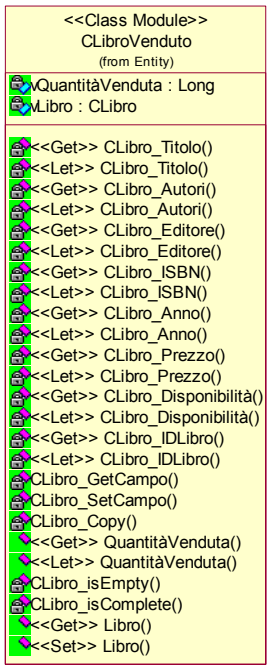
2.9. CLibroVenduto

La classe contiene le informazioni relative ad un libro presente in una determinata vendita; essa eredita dalla classe CLibro metodi e attributi.

Attributi

- *Private vQuantitàVenduta As Long*
  Indica l'ammontare della quantità venduta per il libro in questione.

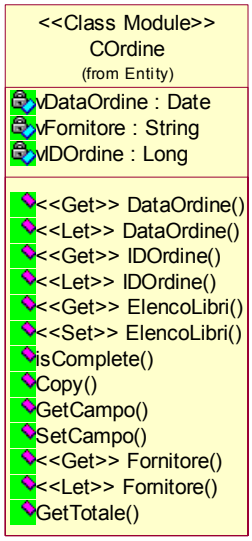
Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.



2.10. COrdine

La classe contiene le informazioni relative ad un generico ordine.

Attributi



- *Private vDataOrdine As Date*
  La data di effettuazione dell'ordine.

- *Private vFornitore As String*
  Il nome del fornitore per l'ordine in questione.

- *Private vIDOrdine As Long*
  L'ID dell'ordine; consente di identificare l'ordine in maniera univoca.

- *Private vElencoLibri As Collection*
  Lista contente i libri costituenti l'ordine in questione.

Metodi

- *Public Function GetCampo(Index As String) As Variant*
  In base all'indice *Index* passato come argomento, restituisce il valore dell'attributo corrispondente. Il campo Index può assumere i seguenti valori: `DataOrdine`, `Fornitore`, `IDOrdine`

- *Public Function SetCampo(Index As String, value As Variant) As Boolean*
  Assegna il valore *value* al campo identificato dall'indice *Index* restituendo TRUE se l'operazione è andata a buon fine, FALSE altrimenti.

- *Public Function Copy() As COrdine*
  Consente di ottenere il riferimento ad una copia dell'oggetto COrdine.

- *Public Function IsEmpty() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto COrdine sono vuoti.

- *Public Function IsComplete() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto COrdine sono stati riempiti.

- *Public Function GetTotale() As Currency*
  Restituisce il costo totale dell'ordine in questione.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.

2.11. CLibroOrdinato

La classe contiene le informazioni relative ad un libro presente in un determinato ordine; essa eredita dalla classe CLibro metodi e attributi.

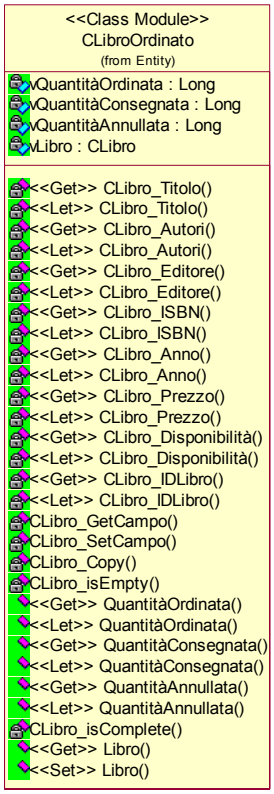
Attributi

- *Private vQuantitàOrdinata As Long*
  Indica l'ammontare della quantità ordinata per il libro in questione.

- *Private vQuantitàConsegnata As Long*
  Indica l'ammontare della quantità consegnata di un libro.

- *Private vQuantitàAnnullata As Long*
  Indica l'ammontare della quantità annullata di un libro.

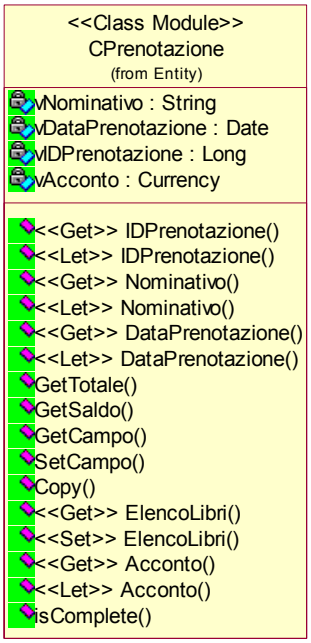
Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.



2.12. CPrenotazione

La classe contiene le informazioni relative ad un generico ordine.

Attributi



- *Private vDataPrenotazione As Date*
  La data di effettuazione della prenotazione.

- *Private vNominativo As String*
  Il nome del cliente che effettua la prenotazione.

- *Private vAcconto As Currency*
  L'ammontare dell'acconto lasciato dal cliente per la prenotazione del libro.

- *Private vIDPrenotazione As Long*
  L'ID della prenotazione; consente di identificare in maniera univoca la prenotazione.

- *Private vElencoLibri As Collection*
  Lista contente i libri costituenti la prenotazione in questione.

Metodi

- *Public Function GetCampo(Index As String) As Variant*
  In base all'indice *Index* passato come argomento, restituisce il valore dell'attributo corrispondente. Il campo Index può assumere i seguenti valori:
  ```
  DataPrenotazione
  Nominativo
  IDPrenotazione
  ```

- *Public Function SetCampo(Index As String, value As Variant) As Boolean*
  Assegna il valore *value* al campo identificato dall'indice *Index* restituendo TRUE se l'operazione è andata a buon fine, FALSE altrimenti.

- *Public Function Copy() As CPrenotazione*
  Consente di ottenere il riferimento ad una copia dell'oggetto CPrenotazione.

- *Public Function IsEmpty() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CPrenotazione sono vuoti.

- *Public Function IsComplete() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CPrenotazione sono stati riempiti.

- *Public Function GetTotale() As Currency*
  Restituisce il costo totale della prenotazione in questione.

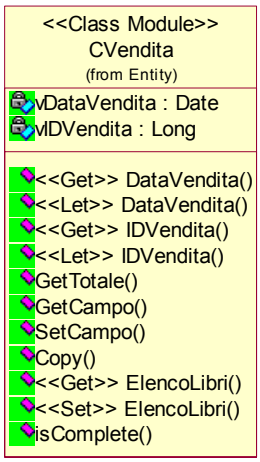
- *Public Function GetSaldo() As Currency*
  Restituisce il saldo ancora da versare per la conferma dell'acquisto.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.

2.13. CVendita

La classe contiene le informazioni relative ad una generica vendita.

Attributi



- *Private vDataVendita As Date*
  La data di effettuazione della vendita.

- *Private vIDVendita As Long*
  L'ID della vendita; consente di identificare in maniera univoca una vendita.

- *Private vElencoLibri As Collection*
  Lista contente i libri costituenti la vendita in questione.

Metodi

- *Public Function GetCampo(Index As String) As Variant*
  In base all'indice *Index* passato come argomento, restituisce il valore dell'attributo corrispondente. Il campo Index può assumere i seguenti valori: `DataVendita`
  `IDVendita`

- *Public Function SetCampo(Index As String, value As Variant) As Boolean*
  Assegna il valore *value* al campo identificato dall'indice *Index* restituendo TRUE se l'operazione è andata a buon fine, FALSE altrimenti.

- *Public Function Copy() As CVendita*
  Consente di ottenere il riferimento ad una copia dell'oggetto CVendita.

- *Public Function IsEmpty() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CVendita sono vuoti.

- *Public Function IsComplete() As Boolean*
  Restituisce TRUE quando tutti gli attributi dell'oggetto CVendita sono stati riempiti.

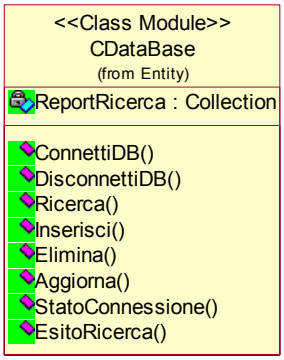
- *Public Function GetTotale() As Currency*
  Restituisce il costo totale della vendita in questione.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.

2.14. CDataBase

La classe fornisce servizi verso i tre sottosistemi Commesso, Magazzino e Responsabile, costituendo l'interfaccia tra di essi e il database; tale classe implementa servizi di connessione, di interrogazione e immagazzinamento. I servizi implementati si appoggiano, a più basso livello ai metodi implementati nelle classi CQuery e CStorage.

Attributi



- *Private ReportRicerca As Collection*
  Variabile che contiene i risultati dell'interrogazione di qualsiasi tipo di ricerca al database.

- *Private ObjConnection As ADODB.Connection*
  Permette di instaurare la connessione verso il database.

Metodi

- *Public Function ConnettiDB() As Boolean*
  Abilita la connessione verso il database.

- *Public Function DisconnettiDB() As Boolean*
  Disabilita la connessione verso il database.

- *Public Function Ricerca(Item As Object, Optional inizio As Date = 0, Optional fine As Date = 0) As Long*
  Cerca nel database l'oggetto passato in ingresso, specificando eventualmente un intervallo temporale di riferimento e restituisce il numero di oggetti trovati con le medesime caratteristiche di quello passato in ingresso.

- *Public Function Inserisci(Item As Object) As Long*
  Inserisce nel database l'oggetto fornito come parametro di ingresso e restituisce l'ID ad esso assegnato.

- *Public Function Elimina(Item As Object) As Long*
  Elimina dal database l'oggetto fornito come parametro di ingresso.

- *Public Function Aggiorna(Item As Object) As Long*
  Aggiorna nel database i campi dell'oggetto passato come parametro di ingresso.

- *Public Function StatoConnessione() As Boolean*
  Restituisce lo stato della connessione: TRUE se è connesso, FALSE altrimenti.

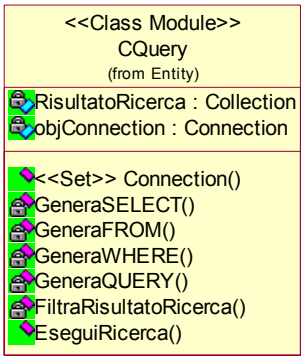
- *Public Function EsitoRicerca() As Collection*
  Restituisce l'esito della ricerca.

Sono inoltre presenti tutti i metodi di lettura e scrittura (set e get) per gli attributi private della classe.

2.15. CQuery

La classe CQuery fornisce servizi di ricerca ed interrogazione al database per le classi CDataBase e CStorage.

Attributi



- *Private RisultatoRicerca As Collection*
  Tale attributo costituisce la lista dei risultati di una ricerca verso il database, opportunamente filtrata.

- *Private ObjConnection As ADODB.Connection*
  Permette di instaurare la connessione verso il database.

- *Private ObjRecordSet As ADODB.RecordSet*
  Contiene il risultato della query eseguita nel database, in termini di recordset.

Metodi

- *Public Property Set Connection(DBConnection As ADODB.Connection)*
  Permette di impostare la connessione verso il database.

- *Private Function GeneraSELECT(Item As Object) As String*
  In base alla tipologia dell'oggetto passato in ingresso (che può essere un libro, un ordine, una prenotazione, ecc…) costruisce la sezione della query SQL relativa al commando di SELECT.

- *Private Function GeneraFROM(Item As Object) As String*
  In base alla tipologia dell'oggetto passato in ingresso (che può essere un libro, un ordine, una prenotazione, ecc…) costruisce la sezione della query SQL relativa al commando di FROM.

- *Private Function GeneraWHERE(Item As Object) As String*
  In base alla tipologia dell'oggetto passato in ingresso (che può essere un libro, un ordine, una prenotazione, ecc…) costruisce la sezione della query SQL relativa al commando di WHERE.

- *Private Function GeneraQUERY(Item As Object) As String*
  In base alla tipologia dell'oggetto passato in ingresso (che può essere un libro, un ordine, una prenotazione, ecc…) costruisce la sezione della query SQL. Al suo interno si trovano le

chiamate verso le funzioni GeneraSELECT, GeneraFROM e GeneraWHERE.

- *Public Function EseguiRicerca(Item As Object, Optional inizio As Date=0, Optional fine As Date=0) As Collection*
  Tale funzione in base alla tipologia dell'oggetto passato in ingresso (che può essere un libro, un ordine, una prenotazione, ecc…) richiama la funzione GeneraQUERY ed effettua interrogazione al database. Il risultato della ricerca viene fornita come Collection; i parametri opzionali *inizio* e *fine* effettuano una ulteriore selezione per oggetti che prevedono come informazione caratterizzante anche dei parametri temporali, come l'oggetto vendita, prenotazione e ordine.

- *Public Function FiltraRisultatoRicerca(Item As Object) As Collection*
  Agisce sul recordset ObjRecordSet, che è stato generato dalla query di selezione, restituendo una Collection di oggetti del tipo passato in ingresso.

2.16. CStorage

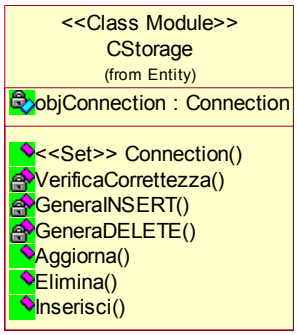
La classe CStorage fornisce funzionalità di immagazzinamento e aggiornamento di informazioni nel database.

Attributi

- *Private ObjConnection As ADODB.Connection*
  Permette di instaurare la connessione verso il database.

- *Private ObjRecordSet As ADODB.RecordSet*
  Contiene il risultato della query eseguita nel database, in termini di recordset.

- *Private ObjQuery As CQuery*
  E' l'istanza dell'oggetto CQuery che fornisce le funzionalità di ricerca prima di eventuali funzioni di immagazzinamento.



Metodi

- *Public Property Set Connection(DBConnection As ADODB.Connection)*
  Permette di impostare la connessione verso il database.

- *Public Function Inserisci(Item As Object) As Long*
  Inserisce nel database l'oggetto fornito come parametro di ingresso e restituisce l'ID ad esso assegnato.

- *Public Function Elimina(Item As Object) As Long*
  Elimina dal database l'oggetto fornito come parametro di ingresso.

- *Public Function Aggiorna(Item As Object) As Long*
  Aggiorna nel database i campi dell'oggetto passato come parametro di ingresso.

- *Private Function GeneraDELETE(Tabella As String, ID As String, ValoreId As Long) As String*
  In base ai parametri passati in ingresso (il nome della tabella del database, il nome dell'ID da eliminare e il suo valore) costruisce la sezione della query SQL relativa al comando DELETE.

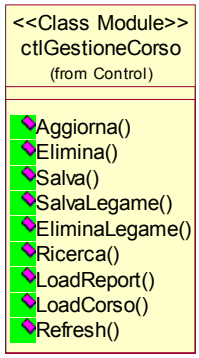
- *Private Function GeneraINSERT(Tabella As String, filtri As String, valori As String) As String*
  In base ai parametri passati in ingresso (il nome della tabella del database, lista dei campi da inserire e i relativi valori) costruisce la sezione della query SQL relativa al comando INSERT.

- *Private Function VerificaCorrettezza(Item As Object) As Boolean*
  Verifica che l'oggetto passato in ingresso abbia tutte le informazioni necessarie e sufficienti per l'inserimento o l'aggiornamento nel database.

2.17. ctlGestioneCorso

Classe di controllo per la gestione dei corsi; questa implementa servizi per la ricerca, la modifica, l'eliminazione ed il salvataggio di corsi; consente inoltre di associare e dissociare i libri adottati da un corso.

Metodi



- *Public Function Aggiorna(Owner As Object) As Long*
  Aggiorna nel database i campi del corso in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function Elimina(Owner As Object) As Long*
  Elimina dal database il corso in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function EliminaLegame(Owner As Object) As Boolean*
  Elimina dal database l'associazione corso – libro adottato; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function Salva(Owner As Object) As Long*
  Inserisce nel database i campi del corso in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function SalvaLegame(Owner As Object) As Boolean*
  Inserisce nel database l'associazione corso – libro adottato; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function Ricerca(tipo As String, Owner As Object) As Long*
  Ricerca il corso in esame nel database, restituendo il numero di elementi trovati se la ricerca è andata a buon fine, 0 altrimenti; *tipo* specifica il tipo di corso a cui mi sto riferendo (CCorso o CCorsoEx), mentre *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub LoadReport(tipo As String, Owner As Object)*
  Permette il caricamento del risultato della ricerca nella form individuata dalla variabile *Owner*. *tipo* indica il tipo di corso da caricare.

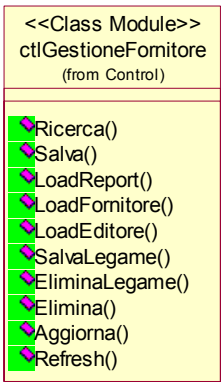
- *Public Sub LoadCorso(id As Long, Owner As Object)*
  Permette il caricamento delle caratteristiche di un particolare corso (individuato da *id*) nella form riferita dalla variabile *Owner*.

- *Public Sub Refresh()*
  Effettua un refresh della form.

2.18. ctlGestioneFornitore

Classe di controllo per la gestione dei fornitori; questa implementa servizi per la ricerca, la modifica, l'eliminazione ed il salvataggio di fornitori; consente inoltre di associare e dissociare gli editori distribuiti da un fornitore.

Metodi



- *Public Function Aggiorna(Owner As Object) As Long*
  Aggiorna nel database i campi del fornitore in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function Elimina(Owner As Object) As Long*
  Elimina dal database il fornitore in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function EliminaLegame(Owner As Object) As Boolean*
  Elimina dal database l'associazione fornitore – editore ; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function Salva(Owner As Object) As Long*
  Inserisce nel database i campi del fornitore in esame; *Owner* si riferisce alla form su cui andrò a lavorare.

- *Public Function SalvaLegame(Owner As Object) As Boolean*
  Inserisce nel database l'associazione fornitore – editore; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function Ricerca(Owner As Object) As Long*
  Ricerca il fornitore in esame nel database, restituendo il numero di elementi trovati se la ricerca è andata a buon fine, 0 altrimenti; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub LoadReport(Owner As Object)*
  Permette il caricamento del risultato della ricerca nella form riferita dalla variabile *Owner*.

- *Public Sub LoadFornitore(ID As Long, Owner As Object)*
  Permette il caricamento delle caratteristiche di un particolare fornitore nella form riferita dalla variabile *Owner*.

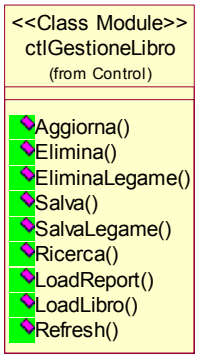
- *Public Sub LoadEditore(Owner As Object)*
  Permette il caricamento della lista degli editori distribuiti dai fornitore nella form riferita dalla variabile *Owner*.

- *Public Sub Refresh()*
  Effettua un refresh della form.

2.19. ctlGestioneLibro

Classe di controllo per la gestione dei libri; questa implementa servizi per la ricerca, la modifica, l'eliminazione ed il salvataggio di libri; consente inoltre di associare e dissociare i corsi che adottano un libro.

Metodi



- *Public Function Aggiorna(Owner As Object) As Long*
  Aggiorna nel database i campi del libro in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function Elimina(Owner As Object) As Long*
  Elimina dal database il libro in esame; *Owner* si riferisce alla form su cui andrò a lavorare.

- *Public Function EliminaLegame(Owner As Object) As Boolean*
  Elimina dal database l'associazione libro – corso; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function Salva(Owner As Object) As Long*
  Inserisce nel database i campi del libro in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function SalvaLegame(Owner As Object) As Boolean*
  Inserisce nel database l'associazione libro – corso; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function Ricerca(tipo As String, Owner As Object) As Long*
  Ricerca il libro in esame nel database, restituendo il numero di elementi trovati se la ricerca è andata a buon fine, 0 altrimenti; *tipo* specifica il tipo di libro a cui mi sto riferendo (CLibro o CLibroEx), mentre *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub LoadReport(tipo As String, Owner As Object)*
  Permette il caricamento del risultato della ricerca nella form riferita dalla variabile *Owner*.

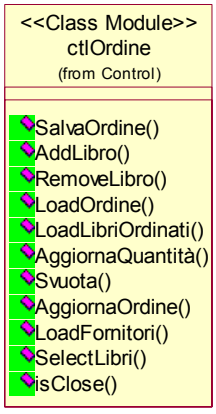
- *Public Sub LoadLibro(ID As Long, Owner As Object)*
  Permette il caricamento delle caratteristiche di un particolare libro nella form riferita dalla variabile *Owner*.

- *Public Sub Refresh()*
  Effettua un refresh della form.

2.20. ctlOrdine

Classe di controllo per la gestione degli ordini; questa implementa servizi per l'aggiornamento, il salvataggio ed la lettura di un ordine.

Metodi



- *Public Function AggiornaOrdine(Owner As Object) As Boolean*
  Aggiorna nel database i campi dell'ordine in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Function SalvaOrdine(Owner As Object) As Long*
  Inserisce nel database i campi dell'ordine in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub LoadLibriOrdinati(Owner As Object)*
  Permette il caricamento di un elenco di libri ordinati nella form riferita dalla variabile *Owner*.

- *Public Sub LoadOrdine(Ordine As COrdine, Owner As Object)*
  Permette il caricamento delle caratteristiche di un particolare ordine nella form riferita dalla variabile *Owner*.

- *Public Sub AddLibro(Libro As CLibro)*
  Permette di aggiungere un libro all'ordine in questione; *Libro* individua il libro da aggiungere.

- *Public Sub RemoveLibro(ID As Long)*
  Permette di rimuovere un libro dall'ordine in esame; *ID* identifica il libro da eliminare fra i libri esistenti nell'ordine.

- *Public Sub AggiornaQuantità(ID As Long, Optional consegnata As Long=-1, Optional ordinate As Long=-1, Optional annullata As Long=-1) As Long*
  Permette di modificare la quantità di un libro in un ordine; in particolare agisce sulla quantità consegnata, ordinata o annullata del libro.

- *Public Sub LoadFornitori(Owner As Object)*
  Permette il caricamento delle caratteristiche dei fornitori disponibili per effettuare un ordine; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub SelectLibri(ID As Long, Owner As Object)*
  Permette di selezionare uno o più libri da un insieme di libri visualizzati nella form riferita da Owner in funzione del fornitore caratterizzato dalla variabile *ID*.

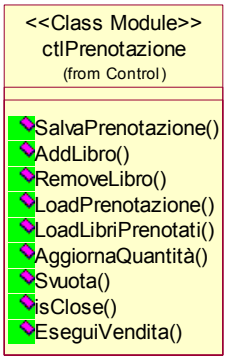
- *Public Sub Svuota()*
  Consente di svuotare i report temporanei.

- *Public Function isClose() as Boolean*
  Restituisce TRUE se l'ordine in questione è stato già salvato, FALSE se ancora aperto.

2.21. ctlPrenotazione

Classe di controllo per la gestione delle prenotazioni; questa implementa servizi per il salvataggio, la lettura, e la vendita di una prenotazione.

Metodi



- *Public Function SalvaPrenotazione(Owner As Object) As Long*
  Inserisce nel database i campi della prenotazione in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub LoadLibriPrenotati(Owner As Object)*
  Permette il caricamento di un elenco di libri prenotati nella form riferita dalla variabile *Owner*.

- *Public Sub LoadPrenotazione(Prenotazione As CPrenotazione, Owner As Object)*
  Permette il caricamento delle caratteristiche di una particolare prenotazione nella form riferita dalla variabile *Owner*.

- *Public Sub AddLibro(Libro As CLibro)*
  Permette di aggiungere un libro alla prenotazione in questione; *Libro* individua il libro da aggiungere.

- *Public Sub RemoveLibro(ID As Long)*
  Permette di rimuovere un libro dalla prenotazione in esame; *ID* identifica il libro da eliminare fra i libri esistenti nella prenotazione.

- *Public Sub AggiornaQuantità(ID As Long, quantità As Long) As Long*
  Permette di modificare la quantità di un libro in una prenotazione.

- *Public Sub EseguiVendita(Owner As Object)*
  Permette il salvataggio della prenotazione nel database; da questo momento l'entità sarà vista non più come prenotazione ma come vendita.

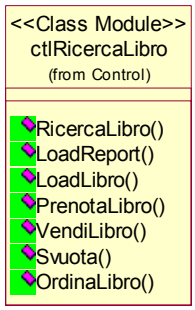
- *Public Sub Svuota()*
  Consente di svuotare i report temporanei.

- *Public Function isClose() as Boolean*
  Restituisce TRUE se l'ordine in questione è stato già salvato, FALSE se ancora aperto.

2.22. ctlRicercaLibro

Classe di controllo per la ricerca dei libri; è caratterizzata da diversi criteri di ricerca quali: libri adottati da un corso, libri venduti, libri prenotati, libri acquistati. Essa prevede anche alcuni servizi di supporto alla vendita, prenotazione e ordinazione di un libro ricercato.

Metodi



- *Public Function RicercaLibro(tipo As String, Owner As Object) As Long*
  Effettua la ricerca di un libro; *tipo* specifica il tipo di libro da ricercare (se appartenente alla classe CLibro o CLibroEx); *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub LoadLibro(ID As Long, Owner As Object)*
  Permette il caricamento delle caratteristiche di un particolare libro nella form riferita dalla variabile *Owner*.

- *Public Sub PrenotaLibro(ID As Long, Owner As Object)*
  Permette di effettuare la prenotazione di un libro; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub VendiLibro(ID As Long, Owner As Object)*
  Permette di effettuare la vendita di un libro; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub OrdinaLibro(ID As Long, Owner As Object)*
  Permette di effettuare l' ordinazione di un libro; *Owner* si riferisce alla form su cui andrà a lavorare.

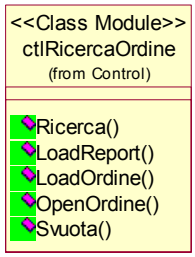
- *Public Sub LoadReport(tipo As String, Owner As Object)*
  Permette il caricamento del risultato della ricerca nella form riferita dalla variabile *Owner*.

- *Public Sub Svuota()*
  Consente di svuotare i report temporanei.

2.23. ctlRicercaOrdine

Classe di controllo per la ricerca degli ordini. Essa si appoggia alla classe CDataBase per i servizi di inserimento, aggiornamento, ecc…

Metodi



- *Public Function Ricerca(Owner As Object) As Long*
  Effettua la ricerca di un ordine; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub LoadOrdine(ID As Long, Owner As Object)*
  Permette il caricamento delle caratteristiche di un particolare ordine identificato da *ID* nella form riferita dalla variabile *Owner*.

- *Public Sub OpenOrdine(ID As Long, Owner As Object)*
  Permette di visualizzare il dettaglio dell'ordine identificato da *ID* nella opportuna form di modifica individuata da *Owner*.

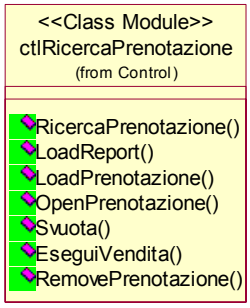
- *Public Sub LoadReport(Owner As Object)*
  Permette il caricamento del risultato della ricerca nella form riferita dalla variabile *Owner*.

- *Public Sub Svuota()*
  Consente di svuotare i report temporanei.

2.24. ctlRicercaPrenotazione

Classe di controllo per la ricerca delle prenotazioni; questa implementa servizi di ricerca, lettura, vendita ed eliminazione di una prenotazione.

Metodi



- *Public Function RicercaPrenotazione (Owner As Object) As Long*
  Effettua la ricerca di una prenotazione; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub LoadPrenotazione(ID As Long, Owner As Object)*
  Permette il caricamento delle caratteristiche di una particolare prenotazione individuata da *ID* nella form riferita dalla variabile *Owner*.

- *Public Sub OpenPrenotazione(ID As Long, Owner As Object)*
  Permette di visualizzare il dettaglio della prenotazione individuata da *ID* nella opportuna form di modifica individuata da *Owner*.

- *Public Sub LoadReport(Owner As Object)*
  Permette il caricamento del risultato della ricerca nella form riferita dalla variabile *Owner*.

- *Public Sub EseguiVendita(Owner As Object)*
  Permette il salvataggio della prenotazione nel database; da questo momento l'entità sarà vista non più come prenotazione ma come vendita.

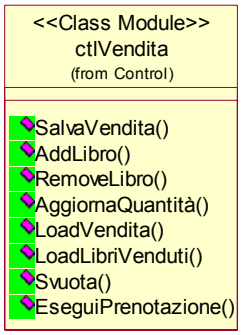
- *Public Sub RemovePrenotazione(ID As Long)*
  Permette la rimozione della prenotazione individuata da *ID* nel database.

- *Public Sub Svuota()*
  Consente di svuotare i report temporanei.

2.25. ctlVendita

Classe di controllo per la gestione delle vendite; questa implementa servizi per il salvataggio, la lettura, e la prenotazione di una vendita.

Metodi



- *Public Function SalvaVendita(Owner As Object) As Long*
  Inserisce nel database i campi della vendita in esame; *Owner* si riferisce alla form su cui andrà a lavorare.

- *Public Sub LoadLibriVenduti(Owner As Object)*
  Permette il caricamento di un elenco di libri venduti nella form riferita dalla variabile *Owner*.

- *Public Sub LoadVendita(Vendita As CVendita, Owner As Object)*
  Permette il caricamento delle caratteristiche di una particolare vendita nella form riferita dalla variabile *Owner*.

- *Public Sub AddLibro(Libro As CLibro)*
  Permette di aggiungere un libro alla vendita in questione; *Libro* individua il libro da aggiungere.

- *Public Sub RemoveLibro(ID As Long)*
  Permette di rimuovere un libro dalla prenotazione in esame; *ID* identifica il libro da rimuovere nella lista di libri appartenenti ala vendita.

- *Public Sub AggiornaQuantità(ID As Long, quantità As Long) As Long*
  Permette di modificare la quantità di un libro in una vendita; *ID* identifica il libro in questione, *quantità* identifica il nuovo valore.

- *Public Sub EseguiPrenotazione(Owner As Object)*
  Permette il salvataggio della vendita nel database; da questo momento l'entità sarà vista non più come vendita ma come prenotazione.

- *Public Sub Svuota()*
  Consente di svuotare i report temporanei.

Appendice

Progettazione del DataBase

Indice

1. Introduzione

E' stata realizzata una base di dati condivisa, il cui accesso è gestito tramite un DBMS (Data Base Management System) sfruttato dal software BOOKBASE.
Si è progettato un archivio contenente tutti i dati di interesse per l'esercizio del committente, dai dati che caratterizzano i libri ai dati fondamentali per un buon supporto alle decisioni.

La progettazione della struttura delle tabelle costituenti il database è stata condotta con l'ausilio di Microsoft Access®, mentre il diagramma relazionale (ERD), di seguito riportato, è stato realizzato con Microsoft Visio®.

2. Progetto Concettuale

2.1. Descrizione dell'archivio

Si vuole realizzare un archivio contenente tutti i dati di maggiore interesse per l'esercizio del committente e più precisamente i corsi delle diverse facoltà/indirizzi dell'università locale ed i relativi libri richiesti o consigliati dai docenti delle materie. Si deve tenere traccia degli ordini, delle giacenze di magazzino, delle eventuali prenotazioni dei clienti, dei diversi fornitori a disposizione e di una history degli acquisti e delle vendite relativi all'esercizio:

- Ogni libro è caratterizzato da un titolo, da autori, da un editore, da un anno di pubblicazione, da un ISBN, da un prezzo e da una quantità (disponibilità).
- Ogni libro è pubblicato da un editore.
- Ogni editore è caratterizzato da un nome.
- Ogni libro è relativo ad un Corso e può essere richiesto o consigliato.
- Ogni corso è caratterizzato da un docente, da una materia, da una facoltà, da un indirizzo, da una data di inizio e una data di fine corso e da un numero medio di studenti partecipanti.
- Ogni ordine è caratterizzato da una data.
- Ogni ordine contiene dei libri caratterizzati da una quantità ordinata, una quantità consegnata e una quantità annullata.

- Ogni fornitore è caratterizzato da una ragione sociale, da una partita IVA, da un indirizzo (sede) , una città, un CAP e un telefono.
- Ogni fornitore distribuisce diversi editori.
- Ogni ordine è fatto ad un fornitore.
- Ogni prenotazione è caratterizzata da un nominativo del cliente, da una data e da un acconto.
- Ogni prenotazione comprende libri secondo una certa quantità.
- Ogni vendita è caratterizzata da una data.
- Ogni vendita comprende dei libri secondo una certa quantità.

2.2. Identificazione elementi dell'ERD

Le **entità** sono in genere rappresentate da sostantivi e sono visualizzate in **rosso**: libro, corso, ordine, fornitore, editore, prenotazione e vendita.
Gli **attributi** sono identificabili come proprietà delle entità e sono visualizzati in **verde**.
Le **relazioni** sono verbi che collegano più entità e sono visualizzate in **blu**. Nel nostro caso:

- Per l'entità ***libro*** gli attributi sono: Titolo, Autori, Editore, Anno di pubblicazione, ISBN, Prezzo e Disponibilità.

- L'entità ***corso*** è in relazione di tipo 'è_relativo_a' con l'entità libro; e i suoi attributi sono: docente, materia, facoltà, indirizzo, inizio, fine e studenti.

- L'entità ***ordine*** è in relazione di tipo 'contiene' con l'entità libro; e il suo attributo è data.
  Si noti che la relazione 'contiene' è caratterizzata dagli attributi quantità richiesta, quantità consegnata e quantità annullata. Infatti un ordine presenta un numero di copie ordinate di uno o più libri, un numero di copie pervenute e le eventuali copie non pervenute che sono state disdette.

- L'entità ***fornitore*** è in relazione di tipo 'è_fatto_ad_un' con l'entità ordine e i suoi attributi sono: ragione sociale, partita IVA, indirizzo, città, CAP e telefono.

- L'entità ***editore*** è in relazione di tipo 'è_pubblicato_da' con l'entità libro ed è in relazione di tipo 'distribuisce' con l'entità fornitore e il suo unico attributo è il nome.

- L'entità ***prenotazione*** è in relazione di tipo 'comprende' con l'entità libro e i suoi attributi sono: nominativo, data e acconto.
  Si noti che la relazione 'comprende' è caratterizzata dall'attributo quantità. Infatti una prenotazione presenta il numero di copie ordinate.

- L'entità ***vendita*** è in relazione di tipo 'comprende' con l'entità libro e il suo unico attributo è data.
  Si noti che la relazione 'comprende' è caratterizzata dall'attributo quantità. Infatti una vendita presenta il numero di copie vendute.

3. Progetto Logico

3.1. Diagramma Entità – Relazione (ERD)

Dalla sezione relativa al progetto concettuale deriva il diagramma ERD riportato in fig.1:

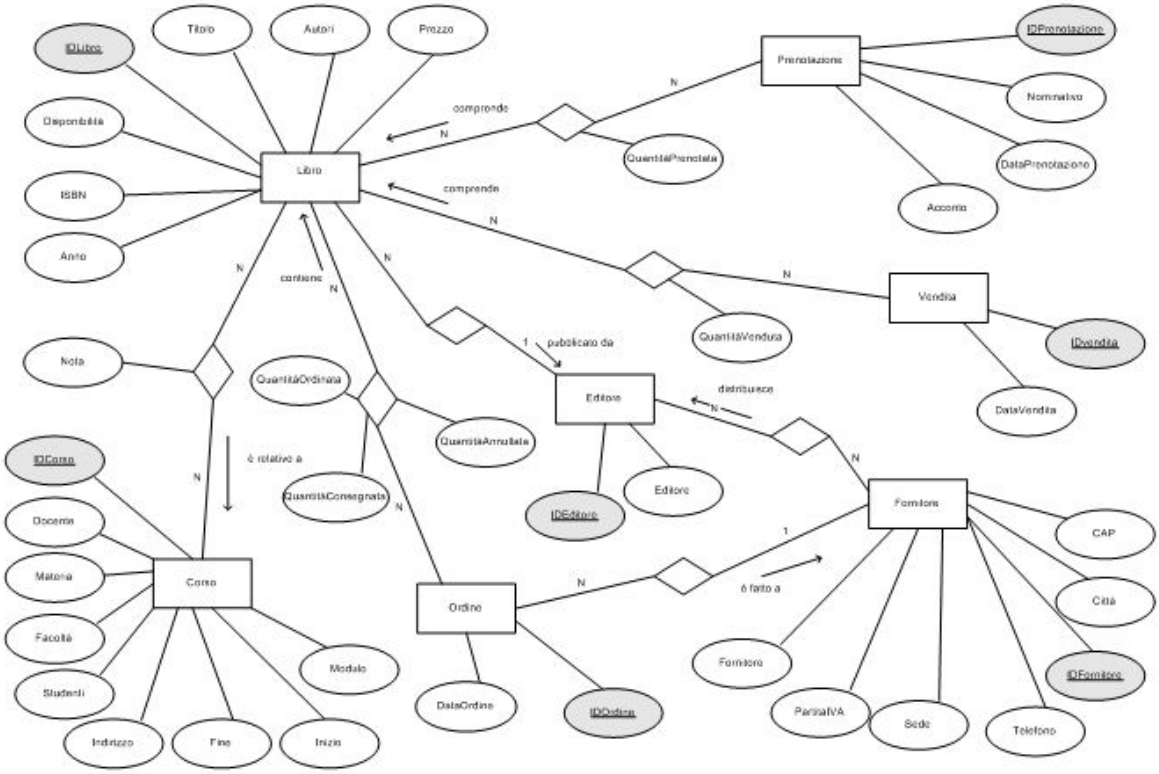


Figura 1 - ERD

4. Descrizione delle Tabelle

4.1. TCorso

La tabella riporta tutti i corsi delle diverse facoltà/indirizzi dell'università locale che possono risultare "convenienti" alla libreria, in termini di possibile guadagno derivante dalla vendita dei libri in essi richiesti o consigliati:

Campo	***Tipo***	***Vincoli***
IDCorso	Contatore	NOT NULL, Richiesto
Docente	Testo(50)	Richiesto
Materia	Testo(50)	Richiesto
Modulo	Testo(10)	Valore predefinito = Unico
Facoltà	Testo(50)	
Indirizzo	Testo(50)	
Inizio	Data/Ora	
Fine	Data/Ora	
Studenti	Intero	> 0, Richiesto

4.2. TLibro

La tabella riporta tutti i libri richiesti o consigliati dai docenti dei corsi riportati nella tabella "Corso":

Campo	***Tipo***	***Vincoli***
IDLibro	Contatore	NOT NULL, Richiesto
IDEditore	Intero lungo	Richiesto
ISBN	Testo(13)	Richiesto
Autori	Testo(50)	Richiesto
Titolo	Testo(50)	Richiesto
Anno	Testo(4)	
Disponibilità	Intero	Richiesto
Prezzo	Valuta	Richiesto

4.3. TEditore

La tabella riporta gli editori relativi ai libri contenuti nella tabella "Libro":

Campo	*Tipo*	*Vincoli*
IDEditore	Contatore	NOT NULL, Richiesto
Editore	Testo(50)	Richiesto

4.4. TCorsoLibro

E' una tabella intermedia che ha lo scopo di realizzare l'opportuna cardinalità della relazione tra le tabelle "Corso" e "Libro" e nel diagramma ERD corrisponde alla relazione 'relativo_a':

Campo	*Tipo*	*Vincoli*
IDCorso	Intero lungo	NOT NULL, Richiesto
IDLibro	Intero lungo	NOT NULL, Richiesto
Nota	Boolean	Richiesto

4.5. TFornitore

La tabella riporta un elenco dei fornitori delle case editrici relative ai libri che tratta l'esercizio:

Campo	*Tipo*	*Vincoli*
IDFornitore	Contatore	NOT NULL, Richiesto
Fornitore	Testo (40)	Richiesto
Partita IVA	Testo(11)	Richiesto
Sede	Testo(30)	
Città	Testo(20)	
CAP	Testo(5)	
Telefono	Testo(20)	Richiesto

4.6. TFornitoreEditore

E' una tabella intermedia che ha lo scopo di realizzare l'opportuna cardinalità della relazione tra le tabelle "Editore" e "Fornitore" e nel diagramma ERD corrisponde alla relazione 'distribuisce':

Campo	*Tipo*	*Vincoli*
IDEditore	Intero lungo	NOT NULL, Richiesto
IDFornitore	Intero lungo	NOT NULL, Richiesto
Zona	Testo(20)	

4.7. TOrdine

La tabella riporta tutti i dettagli relativi agli ordini pendenti e evasi della libreria:

Campo	*Tipo*	*Vincoli*
IDOrdine	Contatore	NOT NULL, Richiesto
IDFornitore	Intero lungo	Richiesto
DataOrdine	Data/Ora	Richiesto (data di emissione)

4.8. TLibroOrdine

E' una tabella intermedia che ha lo scopo di realizzare l'opportuna cardinalità della relazione tra le tabelle "Ordine" e "Libro" e nel diagramma ERD corrisponde alla relazione 'contiene':

Campo	*Tipo*	*Vincoli*
IDOrdine	Intero lungo	NOT NULL, Richiesto
IDLibro	Intero lungo	NOT NULL, Richiesto
QuantitàOrdinata	Intero	>0, Richiesto, Predefinito = 1
QuantitàConsegnata	Intero	Richiesto, Predefinito = 0
QuantitàAnnullata	Intero	Richiesto, Predefinito = 0

4.9. TPrenotazione

La tabella riporta i dettagli relativi alle eventuali richieste di prenotazione dei testi da parte degli studenti o dei docenti, ed è di supporto al personale addetto alla gestione degli ordini:

Campo	*Tipo*	*Vincoli*
IDPrenotazione	Contatore	NOT NULL, Richiesto
Nominativo	Testo(20)	
DataPrenotazione	Data/Ora	Richiesto
Acconto	Valuta	Richiesto

4.10. TLibroPrenotazione

E' una tabella intermedia che ha lo scopo di realizzare l'opportuna cardinalità della relazione tra le tabelle "Prenotazione" e "Libro" e nel diagramma ERD corrisponde alla relazione 'comprende':

Campo	*Tipo*	*Vincoli*
IDPrenotazione	Intero lungo	NOT NULL, Richiesto
IDLibro	Intero lungo	NOT NULL, Richiesto
QuantitàPrenotata	Intero	>0, Richiesto, Predefinito = 1

4.11. TVendita

La tabella riporta i dettagli relativi alle registrazioni di tutte le vendite effettuate dall'esercizio, con le funzioni di "history" dei movimenti e di supporto alle decisioni:

Campo	*Tipo*	*Vincoli*
IDVendita	Contatore	NOT NULL, Richiesto
DataVendita	Data/Ora	Richiesto

4.12. TLibroVendita

E' una tabella intermedia che ha lo scopo di realizzare l'opportuna cardinalità della relazione tra le tabelle "Vendita" e "Libro" e nel diagramma ERD corrisponde alla relazione 'comprende':

Campo	***Tipo***	***Vincoli***
IDVendita	Intero lungo	NOT NULL, Richiesto
IDLibro	Intero lungo	NOT NULL, Richiesto
QuantitàVenduta	Intero	>0, Richiesto

Di seguito si riportano dei diagrammi che mostrano le relazioni fra le tabelle e la loro implementazione in Microsoft Acces:

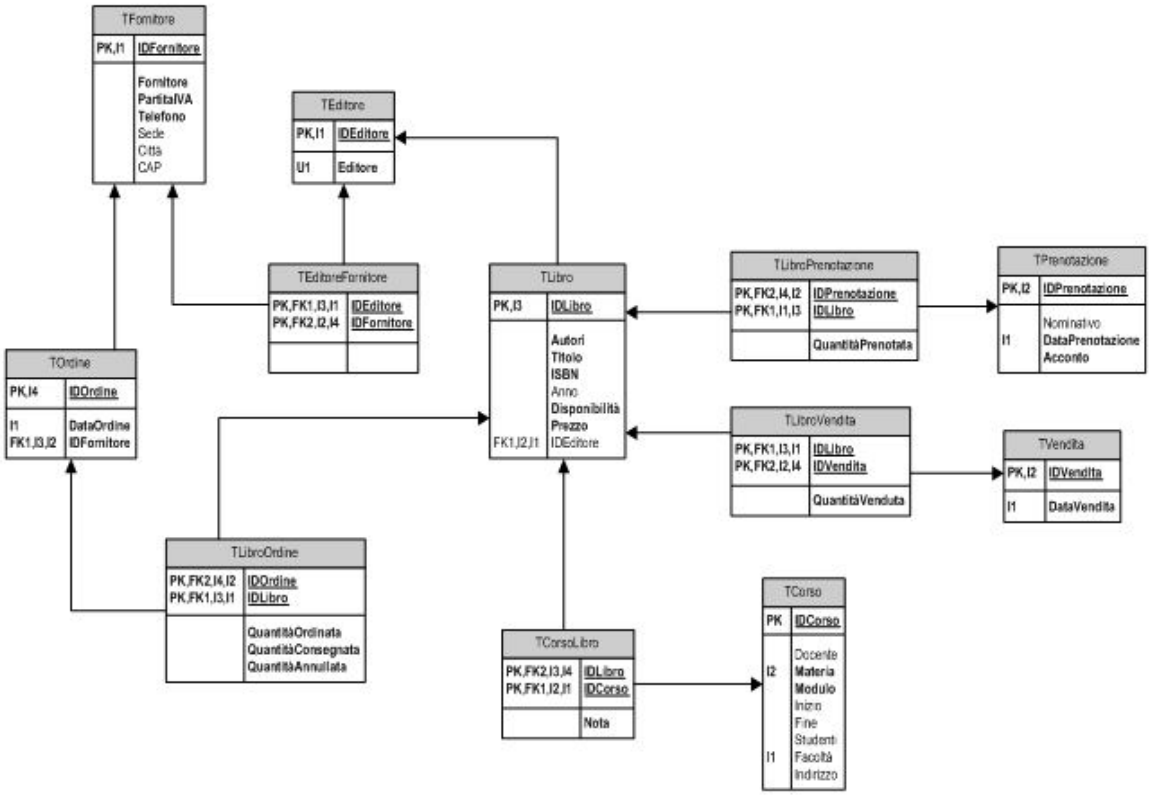


Figura 2 - Diagramma delle relazioni fra le tabelle

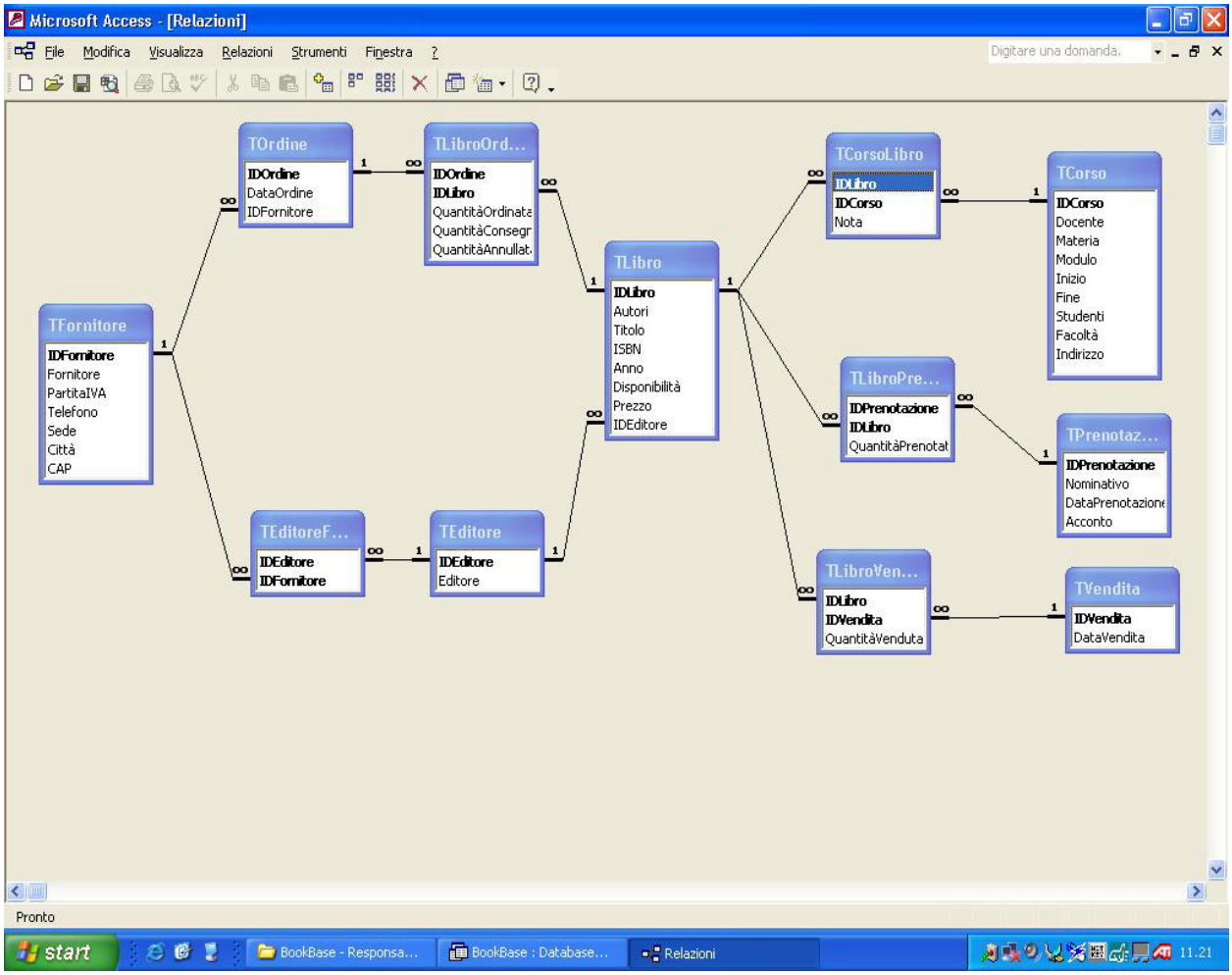


Figura 3 - Struttura e relazioni fra le tabelle in Microsoft Access

Scheletro del Codice Sorgente

CCorso

```
Option Explicit

'##ModelId=3DA2AAAE00A0
Private vMateria As String

'##ModelId=3DA2AAB6000B
Private vDocente As String

'##ModelId=3DA2AABF01D1
Private vModulo As String

'##ModelId=3DA2AACF0292
Private vIndirizzo As String

'##ModelId=3DA2AAD900E8
Private vFacoltà As String

'##ModelId=3DA2AAE30088
Private vNumeroStudenti As Long

'##ModelId=3DA2AAED0155
Private vInizio As Date

'##ModelId=3DA2AAF602F2
Private vFine As Date

'##ModelId=3DB414280285
Private vIDCorso As Long

'##ModelId=3DAAA7E90065
Public Property Get Materia() As String

End Property

'##ModelId=3DAAA7F60000
Public Property Let Materia(item As String)

End Property

'##ModelId=3DAAA809026A
Public Property Get Docente() As String

End Property

'##ModelId=3DAAA815039E
Public Property Let Docente(item As String)

End Property

'##ModelId=3DAAA82B025F
Public Property Get Modulo() As String

End Property

'##ModelId=3DAAA834033E
Public Property Let Modulo(item As String)

End Property

'##ModelId=3DAAA84702CD
```

```
Public Property Get Indirizzo() As String

End Property

'##ModelId=3DAAA851039A
Public Property Let Indirizzo(item As String)

End Property

'##ModelId=3DAAA86402ED
Public Property Get Facoltà() As String

End Property

'##ModelId=3DAAA86F028F
Public Property Let Facoltà(item As String)

End Property

'##ModelId=3DAAA887027F
Public Property Get NumeroStudenti() As Long

End Property

'##ModelId=3DAAA8960122
Public Property Let NumeroStudenti(item As Long)

End Property

'##ModelId=3DAAA8B000B1
Public Property Get inizio() As Date

End Property

'##ModelId=3DAAA8BA02E7
Public Property Let inizio(item As Date)

End Property

'##ModelId=3DAAA8CA0177
Public Property Get fine() As Date

End Property

'##ModelId=3DAAA8D3024C
Public Property Let fine(item As Date)

End Property

'##ModelId=3DB4143B01C5
Public Property Get IDCorso() As Long

End Property

'##ModelId=3DB41453010B
Public Property Let IDCorso(item As Long)

End Property

'##ModelId=3DAAA8EB03B9
Public Function GetCampo(index As String) As Variant

End Function
```

```
'##ModelId=3DAAA8FF01C3
Public Function SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DAAA91B03AE
Public Function Copy() As CCorso

End Function

'##ModelId=3DB41CB60220
Public Function isEmpty() As Boolean

End Function

'##ModelId=3DDE2416022E
Public Function isComplete() As Boolean

End Function
```

CCorsoEx

```
Option Explicit

'##ModelId=3BC5B3760208
Implements CCorso

'##ModelId=3BC9B7840046
Private vCorso As CCorso

'##ModelId=3BCAA53B021C
Private vNota As String

'##ModelId=3BC5B34E01A4
Private vLibriAdottati As Collection

'##ModelId=3DAABCDB00EA
Private Property Get CCorso_Materia() As String

End Property

'##ModelId=3DAABCDB011C
Private Property Let CCorso_Materia(item As String)

End Property

'##ModelId=3DAABCDB0194
Private Property Get CCorso_Docente() As String

End Property

'##ModelId=3DAABCDB01C6
Private Property Let CCorso_Docente(item As String)

End Property

'##ModelId=3DAABCDB023F
Private Property Get CCorso_Modulo() As String
```

```
End Property

'##ModelId=3DAABCDB027B
Private Property Let CCorso_Modulo(item As String)

End Property

'##ModelId=3DAABCDB02F3
Private Property Get CCorso_Indirizzo() As String

End Property

'##ModelId=3DAABCDB032F
Private Property Let CCorso_Indirizzo(item As String)

End Property

'##ModelId=3DAABCDB03B1
Private Property Get CCorso_Facoltà() As String

End Property

'##ModelId=3DAABCDC0005
Private Property Let CCorso_Facoltà(item As String)

End Property

'##ModelId=3DAABCDC0087
Private Property Get CCorso_NumeroStudenti() As Long

End Property

'##ModelId=3DAABCDC00C3
Private Property Let CCorso_NumeroStudenti(item As Long)

End Property

'##ModelId=3DAABCDC0150
Private Property Get CCorso_Inizio() As Date

End Property

'##ModelId=3DAABCDC018C
Private Property Let CCorso_Inizio(item As Date)

End Property

'##ModelId=3DAABCDC0218
Private Property Get CCorso_Fine() As Date

End Property

'##ModelId=3DAABCDC025E
Private Property Let CCorso_Fine(item As Date)

End Property

'##ModelId=3DAABCDC02EA
Private Function CCorso_GetCampo(index As String) As Variant

End Function

'##ModelId=3DAABCDC036C
```

```
Private Function CCorso_SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DAABCDD0043
Private Function CCorso_Copy() As CCorso

End Function

'##ModelId=3DAAC200035B
Public Property Get LibriAdottati() As Collection

End Property

'##ModelId=3DAAC21E0390
Public Property Set LibriAdottati(item As Collection)

End Property

'##ModelId=3BCAA5500014
Public Property Get Nota() As String

End Property

'##ModelId=3BCAA55E0276
Public Property Let Nota(item As String)

End Property

'##ModelId=3DB416D10112
Private Property Get CCorso_IDCorso() As Long

End Property

'##ModelId=3DB416D10114
Private Property Let CCorso_IDCorso(item As Long)

End Property

'##ModelId=3DB41F6F0024
Private Function CCorso_isEmpty() As Boolean

End Function

'##ModelId=3DDE2616020C
Private Function CCorso_isComplete() As Boolean

End Function

'##ModelId=3DE933C40053
Public Property Get Corso() As CCorso

End Property

'##ModelId=3DE9342C014D
Public Property Set Corso(tmpCorso As CCorso)

End Property
```

CDataBase

```
Option Explicit

'##ModelId=3DAA915102C1
Private ReportRicerca As Collection

'##ModelId=3DAA92F501DF
Private objConnection As Connection

'##ModelId=3DAA94D802AB
Public Function ConnettiDB() As Boolean

End Function

'##ModelId=3DAA94E800F6
Public Function DisconnettiDB() As Boolean

End Function

'##ModelId=3DAA95600076
Public Function Ricerca(item As Object, Optional inizio As Date = 0, Optional
fine As Date = 0) As Long

End Function

'##ModelId=3DAA95C600D7
Public Function Inserisci(item As Object) As Long

End Function

'##ModelId=3DAA962F006A
Public Function Elimina(item As Object) As Long

End Function

'##ModelId=3DAA96540258
Public Function Aggiorna(item As Object) As Long

End Function

'##ModelId=3DAA96B9018B
Public Function StatoConnessione() As Boolean

End Function

'##ModelId=3DAA9745038B
Public Function EsitoRicerca() As Collection

End Function
```

CFornitore

```
Option Explicit

'##ModelId=3BC2CB7E02D0
Private vRagioneSociale As String

'##ModelId=3BC2CBE8029E
Private vPartitaIVA As String
```

```
'##ModelId=3BC2CBF6030C
Private vTelefono As String

'##ModelId=3BC2CC150140
Private vSede As String

'##ModelId=3BC2CC1D030C
Private vCittà As String

'##ModelId=3BC2CC26026C
Private vCAP As String

'##ModelId=3DBEA46801CF
Private vIDFornitore As Long

'##ModelId=3DC14809001C
Private vElencoEditori As Collection

'##ModelId=3DBEA4A00085
Public Property Get RagioneSociale() As String

End Property

'##ModelId=3DBEA4B30050
Public Property Let RagioneSociale(item As String)

End Property

'##ModelId=3DBEA4D30101
Public Property Get PartitaIVA() As String

End Property

'##ModelId=3DBEA4E20184
Public Property Let PartitaIVA(item As String)

End Property

'##ModelId=3DBEA4FB00EA
Public Property Get Telefono() As String

End Property

'##ModelId=3DBEA50600E6
Public Property Let Telefono(item As String)

End Property

'##ModelId=3DBEA51C001F
Public Property Get Sede() As String

End Property

'##ModelId=3DBEA5290295
Public Property Let Sede(item As String)

End Property

'##ModelId=3DBEA53D03A2
Public Property Get Città() As String

End Property
```

```
'##ModelId=3DBEA54A0238
Public Property Let Città(item As String)

End Property

'##ModelId=3DBEA560023A
Public Property Get CAP() As String

End Property

'##ModelId=3DBEA56B027C
Public Property Let CAP(item As String)

End Property

'##ModelId=3DBEA57D0187
Public Property Get IDFornitore() As Long

End Property

'##ModelId=3DBEA58B037C
Public Property Let IDFornitore(item As Long)

End Property

'##ModelId=3DBEA5B001DB
Public Function isEmpty() As Boolean

End Function

'##ModelId=3DBEA5C10095
Public Function isComplete() As Boolean

End Function

'##ModelId=3DBEA5D30358
Public Function GetCampo(index As String) As Variant

End Function

'##ModelId=3DBEA5E7003F
Public Function SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DBEA614015C
Public Function Copy() As CFornitore

End Function

'##ModelId=3DBEABC60145
Public Property Get ElencoEditori() As Collection

End Property

'##ModelId=3DBEABEB03B5
Public Property Set ElencoEditori(item As Collection)

End Property
```

CLibro

```
Option Explicit

'##ModelId=3BC078D001FE
Private vTitolo As String

'##ModelId=3BC078D800C8
Private vAutori As String

'##ModelId=3BC078E10140
Private vEditore As String

'##ModelId=3BC078E9000A
Private vISBN As String

'##ModelId=3BC078F201CC
Private vAnno As String

'##ModelId=3BC079030064
Private vPrezzo As Currency

'##ModelId=3BC0790A019A
Private vDisponibilità As Long

'##ModelId=3DAAA4B10331
Private vIDLibro As Long

'##ModelId=3DAAA4F703C8
Public Property Get Titolo() As String

End Property

'##ModelId=3DAAA50903B0
Public Property Let Titolo(item As String)

End Property

'##ModelId=3DAAA5250189
Public Property Get Autori() As String

End Property

'##ModelId=3DAAA5320016
Public Property Let Autori(item As String)

End Property

'##ModelId=3DAAA5530154
Public Property Get Editore() As String

End Property

'##ModelId=3DAAA55F02FF
Public Property Let Editore(item As String)

End Property

'##ModelId=3DAAA5730127
Public Property Get ISBN() As String

End Property
```

```
'##ModelId=3DAAA57D010E
Public Property Let ISBN(item As String)

End Property

'##ModelId=3DAAA58F01DC
Public Property Get Anno() As String

End Property

'##ModelId=3DAAA59801FD
Public Property Let Anno(item As String)

End Property

'##ModelId=3DAAA5AD0302
Public Property Get Prezzo() As Currency

End Property

'##ModelId=3DAAA5B80339
Public Property Let Prezzo(item As Currency)

End Property

'##ModelId=3DAAA5CE0010
Public Property Get Disponibilità() As Long

End Property

'##ModelId=3DAAA5DC0024
Public Property Let Disponibilità(item As Long)

End Property

'##ModelId=3DAAA613015A
Public Property Get IDLibro() As Long

End Property

'##ModelId=3DAAA62100CE
Public Property Let IDLibro(item As Long)

End Property

'##ModelId=3DAAA6A4004A
Public Function GetCampo(index As String) As Variant

End Function

'##ModelId=3DAAA6E3034D
Public Function SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DAAA7400134
Public Function Copy() As CLibro

End Function

'##ModelId=3DB41CA702B5
Public Function isEmpty() As Boolean
```

```
End Function

'##ModelId=3DDE23DD005F
Public Function isComplete() As Boolean

End Function
```

CLibroAnalisiVenduto

```
Option Explicit

'##ModelId=3BC2CF8503AC
Implements CLibro

'##ModelId=3BC2CF4A0082
Private vQuantitàVenduta As Long

'##ModelId=3BC2CF5501B8
Private vPercentualeVendite As Long

'##ModelId=3DE65A3500C9
Private vLibro As CLibro

'##ModelId=3BCAC598006E
Private Property Get CLibro_Titolo() As String

End Property

'##ModelId=3BCAC598014A
Private Property Let CLibro_Titolo(item As String)

End Property

'##ModelId=3BCAC5980370
Private Property Get CLibro_Autori() As String

End Property

'##ModelId=3BCAC59900A0
Private Property Let CLibro_Autori(item As String)

End Property

'##ModelId=3BCAC599028A
Private Property Get CLibro_Editore() As String

End Property

'##ModelId=3BCAC59903A2
Private Property Let CLibro_Editore(item As String)

End Property

'##ModelId=3BCAC59A01E0
Private Property Get CLibro_ISBN() As String

End Property

'##ModelId=3BCAC59A02EE
```

```
Private Property Let CLibro_ISBN(item As String)

End Property

'##ModelId=3BCAC59B015E
Private Property Get CLibro_Anno() As String

End Property

'##ModelId=3BCAC59B0276
Private Property Let CLibro_Anno(item As String)

End Property

'##ModelId=3BCAC59C00E6
Private Property Get CLibro_Prezzo() As Currency

End Property

'##ModelId=3BCAC59C0230
Private Property Let CLibro_Prezzo(item As Currency)

End Property

'##ModelId=3BCAC59D00AA
Private Property Get CLibro_Disponibilità() As Long

End Property

'##ModelId=3BCAC59D01F4
Private Property Let CLibro_Disponibilità(item As Long)

End Property

'##ModelId=3BCAC59E00A0
Private Property Get CLibro_IDLibro() As Long

End Property

'##ModelId=3BCAC59E01EA
Private Property Let CLibro_IDLibro(item As Long)

End Property

'##ModelId=3BCAC59F008C
Private Function CLibro_GetCampo(index As String) As Variant

End Function

'##ModelId=3BCAC59F02EE
Private Function CLibro_SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3BCAC5A002E4
Private Function CLibro_Copy() As CLibro

End Function

'##ModelId=3DBE6D7A016F
Private Function CLibro_isEmpty() As Boolean

End Function
```

```
'##ModelId=3DBEA95F0277
Public Property Get QuantitàVenduta() As Long

End Property

'##ModelId=3DBEA970016D
Public Property Let QuantitàVenduta(item As Long)

End Property

'##ModelId=3DBEA987036F
Public Property Get PercentualeVendite() As Long

End Property

'##ModelId=3DBEA99F03D7
Public Property Let PercentualeVendite(item As Long)

End Property

'##ModelId=3DDE25C50007
Private Function CLibro_isComplete() As Boolean

End Function

'##ModelId=3DE65A8002BB
Public Property Get Libro() As CLibro

End Property

'##ModelId=3DE65AA10313
Public Property Set Libro(tmpLibro As CLibro)

End Property
```

CLibroEx

```
Option Explicit

'##ModelId=3BC5B529005A
Implements CLibro

'##ModelId=3BC9B7730096
Private vLibro As CLibro

'##ModelId=3BCAA4E201F4
Private vNota As String

'##ModelId=3BC5B5590028
Private vCorsiAdottati As Collection

'##ModelId=3DAABCD803DF
Private Property Get CLibro_Titolo() As String

End Property

'##ModelId=3DAABCD90029
Private Property Let CLibro_Titolo(item As String)
```

```
End Property

'##ModelId=3DAABCD90097
Private Property Get CLibro_Autori() As String

End Property

'##ModelId=3DAABCD900D3
Private Property Let CLibro_Autori(item As String)

End Property

'##ModelId=3DAABCD90141
Private Property Get CLibro_Editore() As String

End Property

'##ModelId=3DAABCD9017D
Private Property Let CLibro_Editore(item As String)

End Property

'##ModelId=3DAABCD901EC
Private Property Get CLibro_ISBN() As String

End Property

'##ModelId=3DAABCD90228
Private Property Let CLibro_ISBN(item As String)

End Property

'##ModelId=3DAABCD902A0
Private Property Get CLibro_Anno() As String

End Property

'##ModelId=3DAABCD902DC
Private Property Let CLibro_Anno(item As String)

End Property

'##ModelId=3DAABCD9035E
Private Property Get CLibro_Prezzo() As Currency

End Property

'##ModelId=3DAABCD9039A
Private Property Let CLibro_Prezzo(item As Currency)

End Property

'##ModelId=3DAABCDA0034
Private Property Get CLibro_Disponibilità() As Long

End Property

'##ModelId=3DAABCDA0070
Private Property Let CLibro_Disponibilità(item As Long)

End Property

'##ModelId=3DAABCDA00FD
```

```
Private Property Get CLibro_IDLibro() As Long

End Property

'##ModelId=3DAABCDA0139
Private Property Let CLibro_IDLibro(item As Long)

End Property

'##ModelId=3DAABCDA01C5
Private Function CLibro_GetCampo(index As String) As Variant

End Function

'##ModelId=3DAABCDA023D
Private Function CLibro_SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DAABCDA02FB
Private Function CLibro_Copy() As CLibro

End Function

'##ModelId=3DAAC10B03C7
Public Property Get CorsiAdottati() As Collection

End Property

'##ModelId=3DAAC1510088
Public Property Set CorsiAdottati(corsi As Collection)

End Property

'##ModelId=3BCAA4F700D2
Public Property Get Nota() As String

End Property

'##ModelId=3BCAA5090294
Public Property Let Nota(item As String)

End Property

'##ModelId=3DB41F830127
Private Function CLibro_isEmpty() As Boolean

End Function

'##ModelId=3DDE257801A1
Private Function CLibro_isComplete() As Boolean

End Function

'##ModelId=3DE65AF2011A
Public Property Get Libro() As CLibro

End Property

'##ModelId=3DE65B0203E4
Public Property Set Libro(tmpLibro As CLibro)

End Property
```

CLibroHistory

```
Option Explicit

'##ModelId=3BC2CF190384
Implements CLibro

'##ModelId=3BC2CEB80168
Private vQuantitàVenduta As Long

'##ModelId=3BC2CECA0398
Private vQuantitàOrdinata As Long

'##ModelId=3DE65B390072
Private vLibro As CLibro

'##ModelId=3BCAC58F00DC
Private Property Get CLibro_Titolo() As String

End Property

'##ModelId=3BCAC58F017C
Private Property Let CLibro_Titolo(item As String)

End Property

'##ModelId=3BCAC58F02C6
Private Property Get CLibro_Autori() As String

End Property

'##ModelId=3BCAC58F0371
Private Property Let CLibro_Autori(item As String)

End Property

'##ModelId=3BCAC5900104
Private Property Get CLibro_Editore() As String

End Property

'##ModelId=3BCAC59001A5
Private Property Let CLibro_Editore(item As String)

End Property

'##ModelId=3BCAC590032A
Private Property Get CLibro_ISBN() As String

End Property

'##ModelId=3BCAC5900399
Private Property Let CLibro_ISBN(item As String)

End Property

'##ModelId=3BCAC591012C
Private Property Get CLibro_Anno() As String

End Property

'##ModelId=3BCAC59101D7
```

```
Private Property Let CLibro_Anno(item As String)

End Property

'##ModelId=3BCAC591038E
Private Property Get CLibro_Prezzo() As Currency

End Property

'##ModelId=3BCAC5920015
Private Property Let CLibro_Prezzo(item As Currency)

End Property

'##ModelId=3BCAC59201CC
Private Property Get CLibro_Disponibilità() As Long

End Property

'##ModelId=3BCAC592023B
Private Property Let CLibro_Disponibilità(item As Long)

End Property

'##ModelId=3BCAC59203B6
Private Property Get CLibro_IDLibro() As Long

End Property

'##ModelId=3BCAC5930079
Private Property Let CLibro_IDLibro(item As Long)

End Property

'##ModelId=3BCAC59301F4
Private Function CLibro_GetCampo(index As String) As Variant

End Function

'##ModelId=3BCAC5930370
Private Function CLibro_SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3BCAC5950136
Private Function CLibro_Copy() As CLibro

End Function

'##ModelId=3DBE6D790362
Private Function CLibro_isEmpty() As Boolean

End Function

'##ModelId=3DBEAD3B0001
Public Property Get QuantitàVenduta() As Long

End Property

'##ModelId=3DBEAD4D0269
Public Property Let QuantitàVenduta(item As Long)

End Property
```

```
'##ModelId=3DBEAD660251
Public Property Get QuantitàOrdinata() As Long

End Property

'##ModelId=3DBEADCD01B9
Public Property Let QuantitàOrdinata(item As Long)

End Property

'##ModelId=3DDE259002DC
Private Function CLibro_isComplete() As Boolean

End Function

'##ModelId=3DE65B4500F2
Public Property Get Libro() As CLibro

End Property

'##ModelId=3DE65B5900FA
Public Property Set Libro(tmpLibro As CLibro)

End Property
```

CLibroOrdinato

```
Option Explicit

'##ModelId=3BC2C8100014
Implements CLibro

'##ModelId=3BC2C74D008C
Private vQuantitàOrdinata As Long

'##ModelId=3BC2C76700B4
Private vQuantitàConsegnata As Long

'##ModelId=3BC2C7760230
Private vQuantitàAnnullata As Long

'##ModelId=3DB67D8E0204
Private vLibro As CLibro

'##ModelId=3BCAB91203AC
Private Property Get CLibro_Titolo() As String

End Property

'##ModelId=3BCAB9130033
Private Property Let CLibro_Titolo(item As String)

End Property

'##ModelId=3BCAB913010E
Private Property Get CLibro_Autori() As String

End Property
```

```
'##ModelId=3BCAB913017D
Private Property Let CLibro_Autori(item As String)

End Property

'##ModelId=3BCAB9130258
Private Property Get CLibro_Editore() As String

End Property

'##ModelId=3BCAB91302C7
Private Property Let CLibro_Editore(item As String)

End Property

'##ModelId=3BCAB91303A2
Private Property Get CLibro_ISBN() As String

End Property

'##ModelId=3BCAB9140029
Private Property Let CLibro_ISBN(item As String)

End Property

'##ModelId=3BCAB91400FA
Private Property Get CLibro_Anno() As String

End Property

'##ModelId=3BCAB9140169
Private Property Let CLibro_Anno(item As String)

End Property

'##ModelId=3BCAB9140244
Private Property Get CLibro_Prezzo() As Currency

End Property

'##ModelId=3BCAB91402B3
Private Property Let CLibro_Prezzo(item As Currency)

End Property

'##ModelId=3BCAB91403CA
Private Property Get CLibro_Disponibilità() As Long

End Property

'##ModelId=3BCAB9150051
Private Property Let CLibro_Disponibilità(item As Long)

End Property

'##ModelId=3BCAB915012C
Private Property Get CLibro_IDLibro() As Long

End Property

'##ModelId=3BCAB915019B
Private Property Let CLibro_IDLibro(item As Long)
```

```
End Property

'##ModelId=3BCAB9150276
Private Function CLibro_GetCampo(index As String) As Variant

End Function

'##ModelId=3BCAB9150352
Private Function CLibro_SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3BCAB91600E6
Private Function CLibro_Copy() As CLibro

End Function

'##ModelId=3DB671C00190
Private Function CLibro_isEmpty() As Boolean

End Function

'##ModelId=3DB67DCC0068
Public Property Get QuantitàOrdinata() As Long

End Property

'##ModelId=3DB67DE10163
Public Property Let QuantitàOrdinata(item As Long)

End Property

'##ModelId=3DB67DF9029E
Public Property Get QuantitàConsegnata() As Long

End Property

'##ModelId=3DB67E0E0027
Public Property Let QuantitàConsegnata(item As Long)

End Property

'##ModelId=3DB67E24017D
Public Property Get QuantitàAnnullata() As Long

End Property

'##ModelId=3DB67E3801A4
Public Property Let QuantitàAnnullata(item As Long)

End Property

'##ModelId=3DDE259903C5
Private Function CLibro_isComplete() As Boolean

End Function

'##ModelId=3DE65B9102BE
Public Property Get Libro() As CLibro

End Property

'##ModelId=3DE65B9F0173
```

```
Public Property Set Libro(tmpLibro As CLibro)

End Property
```

CLibroPrenotato

```
Option Explicit

'##ModelId=3DA2A5EE0029
Implements CLibro

'##ModelId=3DA2A57A034E
Private vQuantitàPrenotata As Long

'##ModelId=3BC9B73D00A0
Private vLibro As CLibro

'##ModelId=3DAABCD30337
Private Property Get CLibro_Titolo() As String

End Property

'##ModelId=3DAABCD30373
Private Property Let CLibro_Titolo(item As String)

End Property

'##ModelId=3DAABCD4000E
Private Property Get CLibro_Autori() As String

End Property

'##ModelId=3DAABCD4004A
Private Property Let CLibro_Autori(item As String)

End Property

'##ModelId=3DAABCD400CC
Private Property Get CLibro_Editore() As String

End Property

'##ModelId=3DAABCD40108
Private Property Let CLibro_Editore(item As String)

End Property

'##ModelId=3DAABCD40194
Private Property Get CLibro_ISBN() As String

End Property

'##ModelId=3DAABCD401D0
Private Property Let CLibro_ISBN(item As String)

End Property

'##ModelId=3DAABCD4025C
Private Property Get CLibro_Anno() As String
```

```
End Property

'##ModelId=3DAABCD402A3
Private Property Let CLibro_Anno(item As String)

End Property

'##ModelId=3DAABCD4032F
Private Property Get CLibro_Prezzo() As Currency

End Property

'##ModelId=3DAABCD40375
Private Property Let CLibro_Prezzo(item As Currency)

End Property

'##ModelId=3DAABCD50019
Private Property Get CLibro_Disponibilità() As Long

End Property

'##ModelId=3DAABCD50069
Private Property Let CLibro_Disponibilità(item As Long)

End Property

'##ModelId=3DAABCD500FF
Private Property Get CLibro_IDLibro() As Long

End Property

'##ModelId=3DAABCD50146
Private Property Let CLibro_IDLibro(item As Long)

End Property

'##ModelId=3DAABCD501DC
Private Function CLibro_GetCampo(index As String) As Variant

End Function

'##ModelId=3DAABCD50268
Private Function CLibro_SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DAABCD5033A
Private Function CLibro_Copy() As CLibro

End Function

'##ModelId=3DAABEE402F2
Public Property Get QuantitàPrenotata() As Long

End Property

'##ModelId=3DAABF00025C
Public Property Let QuantitàPrenotata(item As Long)

End Property

'##ModelId=3DB41FF10003
```

```
Private Function CLibro_isEmpty() As Boolean

End Function

'##ModelId=3DDE25A20223
Private Function CLibro_isComplete() As Boolean

End Function

'##ModelId=3DE65BD703AF
Public Property Get Libro() As CLibro

End Property

'##ModelId=3DE65BF1018F
Public Property Set Libro(tmpLibro As CLibro)

End Property
```

CLibroVenduto

```
Option Explicit

'##ModelId=3BC2C38200E6
Implements CLibro

'##ModelId=3BC2C3D003A2
Private vQuantitàVenduta As Long

'##ModelId=3BC9B7540118
Private vLibro As CLibro

'##ModelId=3DAABCD60346
Private Property Get CLibro_Titolo() As String

End Property

'##ModelId=3DAABCD6036E
Private Property Let CLibro_Titolo(item As String)

End Property

'##ModelId=3DAABCD603D2
Private Property Get CLibro_Autori() As String

End Property

'##ModelId=3DAABCD70013
Private Property Let CLibro_Autori(item As String)

End Property

'##ModelId=3DAABCD70076
Private Property Get CLibro_Editore() As String

End Property

'##ModelId=3DAABCD7009F
Private Property Let CLibro_Editore(item As String)
```

```
End Property

'##ModelId=3DAABCD70102
Private Property Get CLibro_ISBN() As String

End Property

'##ModelId=3DAABCD7012B
Private Property Let CLibro_ISBN(item As String)

End Property

'##ModelId=3DAABCD70199
Private Property Get CLibro_Anno() As String

End Property

'##ModelId=3DAABCD701B8
Private Property Let CLibro_Anno(item As String)

End Property

'##ModelId=3DAABCD70225
Private Property Get CLibro_Prezzo() As Currency

End Property

'##ModelId=3DAABCD70244
Private Property Let CLibro_Prezzo(item As Currency)

End Property

'##ModelId=3DAABCD702B1
Private Property Get CLibro_Disponibilità() As Long

End Property

'##ModelId=3DAABCD702D0
Private Property Let CLibro_Disponibilità(item As Long)

End Property

'##ModelId=3DAABCD7033D
Private Property Get CLibro_IDLibro() As Long

End Property

'##ModelId=3DAABCD70365
Private Property Let CLibro_IDLibro(item As Long)

End Property

'##ModelId=3DAABCD703C9
Private Function CLibro_GetCampo(index As String) As Variant

End Function

'##ModelId=3DAABCD80045
Private Function CLibro_SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DAABCD800E6
```

```
Private Function CLibro_Copy() As CLibro

End Function

'##ModelId=3DAABF2B01E6
Public Property Get QuantitàVenduta() As Long

End Property

'##ModelId=3DAABF3C0122
Public Property Let QuantitàVenduta(item As Long)

End Property

'##ModelId=3DB41FE70352
Private Function CLibro_isEmpty() As Boolean

End Function

'##ModelId=3DDE25AA017A
Private Function CLibro_isComplete() As Boolean

End Function

'##ModelId=3DE65C2501BC
Public Property Get Libro() As CLibro

End Property

'##ModelId=3DE65C3501DD
Public Property Set Libro(tmpLibro As CLibro)

End Property
```

COrdine

```
Option Explicit

'##ModelId=3BC2C629017C
Private vDataOrdine As Date

'##ModelId=3BC2C63001CC
Private vFornitore As String

'##ModelId=3BC2C6370294
Private vIDOrdine As Long

'##ModelId=3BC5BE15006E
Private vElencoLibri As Collection

'##ModelId=3DB67B620363
Public Property Get DataOrdine() As Date

End Property

'##ModelId=3DB67B7D022B
Public Property Let DataOrdine(item As Date)

End Property
```

```
'##ModelId=3DB67B960340
Public Property Get IDOrdine() As Long

End Property

'##ModelId=3DB67BA70091
Public Property Let IDOrdine(item As Long)

End Property

'##ModelId=3DB67BC50080
Public Property Get ElencoLibri() As Collection

End Property

'##ModelId=3DB67BE7006B
Public Property Set ElencoLibri(item As Collection)

End Property

'##ModelId=3DB67C010271
Public Function isComplete() As Boolean

End Function

'##ModelId=3DB67C2401F9
Public Function Copy() As COrdine

End Function

'##ModelId=3DB67C3100CC
Public Function GetCampo(index As String) As Variant

End Function

'##ModelId=3DB67C4A0013
Public Function SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DB67C6A03BD
Public Property Get Fornitore() As String

End Property

'##ModelId=3DB67C7A0366
Public Property Let Fornitore(item As String)

End Property

'##ModelId=3DB6AE570357
Public Function GetTotale() As Currency

End Function
```

CPrenotazione

```
Option Explicit

'##ModelId=3BC077DF02D0
Private vNominativo As String

'##ModelId=3BC077EA021C
Private vDataPrenotazione As Date

'##ModelId=3BC077F60208
Private vIDPrenotazione As Long

'##ModelId=3BCAA885017C
Private vAcconto As Currency

'##ModelId=3BC5C7AE02BC
Private vElencoLibri As Collection

'##ModelId=3DAABD890285
Public Property Get IDPrenotazione() As Long

End Property

'##ModelId=3DAABDAC01BD
Public Property Let IDPrenotazione(item As Long)

End Property

'##ModelId=3DAABDC30102
Public Property Get Nominativo() As String

End Property

'##ModelId=3DAABDD00223
Public Property Let Nominativo(item As String)

End Property

'##ModelId=3DAABDE30389
Public Property Get DataPrenotazione() As Date

End Property

'##ModelId=3DAABDF702DD
Public Property Let DataPrenotazione(item As Date)

End Property

'##ModelId=3DAABE1600E3
Public Function GetTotale() As Currency

End Function

'##ModelId=3DAABE3E03A8
Public Function GetSaldo() As Currency

End Function

'##ModelId=3DAABE5E0385
Public Function GetCampo(index As String) As Variant
```

```
End Function

'##ModelId=3DAABE7801F2
Public Function SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DAABEA400A1
Public Function Copy() As CPrenotazione

End Function

'##ModelId=3DAAC2810125
Public Property Get ElencoLibri() As Collection

End Property

'##ModelId=3DAAC2BE01EB
Public Property Set ElencoLibri(item As Collection)

End Property

'##ModelId=3BCAA892000A
Public Property Get Acconto() As Currency

End Property

'##ModelId=3BCAA8A203D4
Public Property Let Acconto(item As Currency)

End Property

'##ModelId=3DB555120051
Public Function isComplete() As Boolean

End Function
```

CQuery

```
Option Explicit

'##ModelId=3DAA916A02A9
Private RisultatoRicerca As Collection

'##ModelId=3DAA9908011C
Private objConnection As Connection

'##ModelId=3DAA9395015C
Private objRecordSet As Recordset

'##ModelId=3DAA99230067
Public Property Set Connection(DBConnection As Connection)

End Property

'##ModelId=3DAA98110244
Private Function GeneraSELECT(item As Object) As String

End Function
```

```
'##ModelId=3DAA99830155
Private Function GeneraFROM(item As String) As String

End Function

'##ModelId=3DAA99AA0043
Private Function GeneraWHERE(item As Object) As String

End Function

'##ModelId=3DAA9A200089
Private Function GeneraQUERY(item As Object) As String

End Function

'##ModelId=3DAA9A3C02C4
Private Function FiltraRisultatoRicerca(item As Object) As Collection

End Function

'##ModelId=3DAA9B3E0210
Public Function EseguiRicerca(item As Object, Optional inizio As Date = 0,
Optional fine As Date = 0) As Collection

End Function
```

CStorage

```
Option Explicit

'##ModelId=3DAA9BA00153
Private objConnection As Connection

'##ModelId=3DAA9391025B
Private objRecordSet As Recordset

'##ModelId=3DEF50BE0225
Private objQuery As CQuery

'##ModelId=3DAA9BB001EC
Public Property Set Connection(DBConnection As Connection)

End Property

'##ModelId=3DAA9BED0172
Private Function VerificaCorrettezza(item As Object) As Boolean

End Function

'##ModelId=3DAA9C1E0385
Private Function GeneraINSERT(nomeTabella As String, filtri As String, valori As
String) As String

End Function

'##ModelId=3DAA9C3B0337
Private Function GeneraDELETE(tabella As String, id As String, ValoreID As Long)
As String

End Function
```

```
'##ModelId=3DAA9CE70045
Public Function Aggiorna(item As Object) As Long

End Function

'##ModelId=3DAA9D08029B
Public Function Elimina(item As Object) As Long

End Function

'##ModelId=3DAA9D1A0283
Public Function Inserisci(item As Object) As Long

End Function
```

CVendita

```
Option Explicit

'##ModelId=3BC2C2DC02A8
Private vDataVendita As Date

'##ModelId=3BC2C2E50320
Private vIDVendita As Long

'##ModelId=3BC5C5A103DE
Private vElencoLibri As Collection

'##ModelId=3DAABF840234
Public Property Get DataVendita() As Date

End Property

'##ModelId=3DAABF9503AB
Public Property Let DataVendita(item As Date)

End Property

'##ModelId=3DAABFAC01D7
Public Property Get IDVendita() As Long

End Property

'##ModelId=3DAABFB90045
Public Property Let IDVendita(item As Long)

End Property

'##ModelId=3DAABFC80173
Public Function GetTotale() As Currency

End Function

'##ModelId=3DAABFD40347
Public Function GetCampo(index As String) As Variant

End Function

'##ModelId=3DAABFFC036D
```

```
Public Function SetCampo(index As String, value As Variant) As Boolean

End Function

'##ModelId=3DAAC011007E
Public Function Copy() As CVendita

End Function

'##ModelId=3DAAC324006B
Public Property Get ElencoLibri() As Collection

End Property

'##ModelId=3DAAC33B0187
Public Property Set ElencoLibri(item As Collection)

End Property

'##ModelId=3DB59818001A
Public Function isComplete() As Boolean

End Function
```

ctlGestioneCorso

```
Option Explicit

'##ModelId=3DD8E366029E
Public Function Aggiorna(owner As Object) As Long

End Function

'##ModelId=3DD8E36B01BF
Public Function Elimina(owner As Object) As Long

End Function

'##ModelId=3DD8E36F00F2
Public Function Salva(owner As Object) As Long

End Function

'##ModelId=3DD8E37802A4
Public Function SalvaLegame(owner As Object) As Boolean

End Function

'##ModelId=3DD8E382001D
Public Function EliminaLegame(owner As Object) As Boolean

End Function

'##ModelId=3DD8E38E0269
Public Function Ricerca(item As String, owner As Object) As Long

End Function

'##ModelId=3DD8E393039D
Public Sub LoadReport(tipo As String, owner As Object)
```

```
End Sub

'##ModelId=3DD8E39901CF
Public Sub LoadCorso(id As Long, owner As Object)

End Sub

'##ModelId=3DDFC7F00007
Public Sub Refresh(id As Long)

End Sub
```

ctlGestioneLibro

```
Option Explicit

'##ModelId=3BCAB53202C6
Public Function Aggiorna(owner As Object) As Long

End Function

'##ModelId=3BCAB559024E
Public Function Elimina(owner As Object) As Long

End Function

'##ModelId=3BCAB57D0208
Public Function EliminaLegame(owner As Object) As Boolean

End Function

'##ModelId=3BCAB5DA0122
Public Function Salva(owner As Object) As Long

End Function

'##ModelId=3BCAB5F801B8
Public Function SalvaLegame(owner As Object) As Boolean

End Function

'##ModelId=3BCAB62C02DA
Public Function Ricerca(item As String, owner As Object) As Long

End Function

'##ModelId=3DB6CBD70215
Public Sub LoadReport(tipo As String, owner As Object)

End Sub

'##ModelId=3DB6D7640109
Public Sub LoadLibro(id As Long, owner As Object)

End Sub

'##ModelId=3DDFC779033B
Public Sub Refresh(id As Long)

End Sub
```

ctlPrenotazione

```
Option Explicit

'##ModelId=3DB54ED5022B
Public Function SalvaPrenotazione(owner As Object) As Long

End Function

'##ModelId=3DB550950366
Public Sub AddLibro(Libro As CLibro)

End Sub

'##ModelId=3DB5530E0384
Public Sub RemoveLibro(id As Long)

End Sub

'##ModelId=3DB55C9F02B3
Public Sub LoadPrenotazione(prenotazione As CPrenotazione, owner As Object)

End Sub

'##ModelId=3DB55DB0023D
Public Function LoadLibriPrenotati(owner As Object) As Long

End Function

'##ModelId=3DB56CE502FA
Public Sub AggiornaQuantità(id As Long, quantità As Long)

End Sub

'##ModelId=3DB683EF0324
Public Sub Svuota()

End Sub

'##ModelId=3DE36B150159
Public Function isClose() As Boolean

End Function

'##ModelId=3DE38F2102CD
Public Sub EseguiVendita(owner As Object)

End Sub
```

ctlRicercaLibro

```
Option Explicit

'##ModelId=3DB40C7202D7
Public Function RicercaLibro(tipo As String, owner As Object) As Long

End Function

'##ModelId=3DB42BFD035B
Public Sub LoadReport(tipo As String, owner As Object)
```

```
End Sub

'##ModelId=3DB5230D0398
Public Sub LoadLibro(id As Long, owner As Object)

End Sub

'##ModelId=3DB55FA3005A
Public Sub PrenotaLibro(id As Long, owner As Object)

End Sub

'##ModelId=3DB684AC027C
Public Sub VendiLibro(id As Long, owner As Object)

End Sub

'##ModelId=3DB66EE502A8
Public Sub Svuota()

End Sub

'##ModelId=3DC0ED2D037D
Public Sub OrdinaLibro(id As Long, owner As Object)

End Sub
```

ctlRicercaPrenotazione

```
Option Explicit

'##ModelId=3DB5838C02DE
Public Function RicercaPrenotazione(owner As Object) As Long

End Function

'##ModelId=3DB585EB015A
Public Sub LoadReport(owner As Object)

End Sub

'##ModelId=3DB58A970182
Public Sub LoadPrenotazione(id As Long, owner As Object)

End Sub

'##ModelId=3DB58E370353
Public Sub OpenPrenotazione(id As Long, owner As Object)

End Sub

'##ModelId=3DB683F802F5
Public Sub Svuota()

End Sub

'##ModelId=3DE38FC501C4
Public Sub EseguiVendita(id As Long, owner As Object)
```

```
End Sub

'##ModelId=3DE3969201A9
Public Sub RemovePrenotazione(id As Long)

End Sub
```

ctlVendita

```
Option Explicit

'##ModelId=3DB55BE20239
Public Function SalvaVendita(owner As Object) As Long

End Function

'##ModelId=3DB6668501B4
Public Sub AddLibro(Libro As CLibro)

End Sub

'##ModelId=3DB6669E02F1
Public Sub RemoveLibro(id As Long)

End Sub

'##ModelId=3DB666B403E3
Public Sub AggiornaQuantità(id As Long, quantità As Long)

End Sub

'##ModelId=3DB666E60395
Public Sub LoadVendita(vendita As CVendita, owner As Object)

End Sub

'##ModelId=3DB6671303AD
Public Function LoadLibriVenduti(owner As Object) As Long

End Function

'##ModelId=3DB683FF033C
Public Sub Svuota()

End Sub

'##ModelId=3DE3A1F2039B
Public Sub EseguiPrenotazione(owner As Object)

End Sub
```

ctlOrdine

```
Option Explicit

'##ModelId=3DB6A5A401CE
Public Function SalvaOrdine(owner As Object) As Long
```

```
End Function

'##ModelId=3DB6A5DA0244
Public Sub AddLibro(Libro As CLibro)

End Sub

'##ModelId=3DB6A5F00390
Public Sub RemoveLibro(id As Long)

End Sub

'##ModelId=3DB6A6050033
Public Sub LoadOrdine(ordine As COrdine, owner As Object)

End Sub

'##ModelId=3DB6A62001D6
Public Sub LoadLibriOrdinati(owner As Object)

End Sub

'##ModelId=3DB6A638013A
Public Sub AggiornaQuantità(id As Long, Optional consegnata As Long = -1,
Optional ordinata As Long = -1, Optional annullata As Long = -1)

End Sub

'##ModelId=3DB6A6820014
Public Sub Svuota()

End Sub

'##ModelId=3DB6B8CF0134
Public Function AggiornaOrdine(owner As Object) As Boolean

End Function

'##ModelId=3DD8EF620245
Public Sub LoadFornitori(owner As Object)

End Sub

'##ModelId=3DDA71A402A8
Public Sub SelectLibri(id As Long, owner As Object)

End Sub

'##ModelId=3DE362C800FA
Public Function isClose() As Boolean

End Function
```

ctlRicercaOrdine

```
Option Explicit

'##ModelId=3DB6A23B0205
Public Function Ricerca(owner As Object) As Long
```

```
End Function

'##ModelId=3DB6A28800C5
Public Sub LoadReport(owner As Object)

End Sub

'##ModelId=3DB6A2BF0042
Public Sub LoadOrdine(id As Long, owner As Object)

End Sub

'##ModelId=3DB6A2E30076
Public Sub OpenOrdine(id As Long, owner As Object)

End Sub

'##ModelId=3DB69B0100CE
Public Sub Svuota()

End Sub
```

ctlGestioneFornitore

```
Option Explicit

'##ModelId=3DC1423E033A
Public Function Ricerca(tipo As String, owner As Object) As Long

End Function

'##ModelId=3DC14245011E
Public Function Salva(owner As Object) As Long

End Function

'##ModelId=3DC1437702FE
Public Sub LoadReport(tipo As String, owner As Object)

End Sub

'##ModelId=3DC1473902C5
Public Sub LoadFornitore(id As Long, owner As Object)

End Sub

'##ModelId=3DD8F941023F
Public Sub LoadEditore(owner As Object)

End Sub

'##ModelId=3DD910C10344
Public Function SalvaLegame(owner As Object) As Boolean

End Function

'##ModelId=3DD910D1017A
Public Function EliminaLegame(owner As Object) As Boolean

End Function
```

```
'##ModelId=3DD92CE00223
Public Function Elimina(owner As Object) As Long

End Function

'##ModelId=3DD92D30037D
Public Function Aggiorna(owner As Object) As Long

End Function

'##ModelId=3DDFCD47002A
Public Sub Refresh(id As Long)

End Sub
```