SDD

System Design Document

AEMME

Riferimento	SDD_AEMME
Versione	0.8
Data	28/12/2024
Destinatario	Prof. Carmine Gravino
Presentato da	NC13

Revision History

Data	Versione	Descrizione	Autori
18/11/2024	0.1	Prima stesura	MDL, GG, AR, CS
23/11/2024	0.2	Design goals e trades off	CS, GG
25/11/2024	0.3	Scomposizione sottosistemi	MDL
25/11/2024	0.4	Controllo degli accessi e servizi dei sottosistemi	MDL, AR
10/12/2024	0.5	Mapping hardware/software e persistenza dei dati	AR
20/12/2024	0.6	Design pattern	GG
23/12/2024	0.7	Prima revisione	AR, MDL
28/12/2024	0.8	Seconda revisione	CS, GG

Team members

Nome	Ruolo nel progetto	Acronimo	Informazioni di contatto
Alessandra Raia	Team Member	AR	a.raia7@studenti.unisa.it
Gregorio Garofalo	Team Member	GG	g.garofalo31@studenti.unisa.it
Martina De Lucia	Team Member	MDL	m.delucia18@studenti.unisa.it
Carla Stefanile	Team Member	CS	c.stefanile1@studenti.unisa.it

Sommario

1 Introduzione

1.1 Scopo del sistema

La casa editrice AEMME ha la necessità di espandere e migliorare la propria piattaforma di vendita online, con l'obiettivo di incrementare la visibilità e la competitività nel mercato dell'e-commerce librario. Il sistema dovrà offrire ai clienti un'esperienza di acquisto semplice, intuitiva e veloce, fornendo un ampio catalogo di libri sempre aggiornato e permettendo di personalizzare l'esperienza sulla base delle preferenze espresse dagli utenti.

Il sistema, gestito da tre differenti gestori, permetterà di controllare e ottimizzare il flusso di ordini, aggiornare il catalogo, monitorare disponibilità, promozioni e prezzi, nonché coordinare la distribuzione tra i negozi fisici e l'e-commerce. I clienti potranno visualizzare i libri disponibili, personalizzare la ricerca, gestire carrello e wishlist, ed effettuare ordini in modo sicuro.

Inoltre, al fine di promuovere l'utilizzo e la vendita, il sistema offre la possibilità per i potenziali lettori di usufruire di programmi fedeltà.

1.2 Obiettivi di Design (Design Goals)

Nella presente sezione si andranno a presentare i Design Goals, ovvero le qualità sulle quali il sistema deve essere focalizzato, formalizzati esplicitamente così che qualsiasi importante decisione di design può essere fatta consistentemente seguendo lo stesso insieme di design goal.

Seguendo le linee guida del libro Bernd Bruegge – Object Oriented Software Engineering i design goal sono statti suddivisi nelle seguenti categorie:

- **Performance**: includono i requisiti di spazio e velocità imposti sul sistema.
- **Dependability**: determinano quanto sforzo deve essere speso per minimizzare i fallimenti del sistema (crash, falle di sicurezza) e le loro conseguenze.
- **Maintenance**: determina quanto sforzo è necessario per modificare il sistema dopo il suo rilascio.
- **End User**: includono qualità che sono desiderabili dal punto di vista dell'utente, ma che non sonostate coperte dai criteri di Performance e Dependability.

Ciascun design goal è descritto da:

- **Rank**, che ne specifica un valore di priorità compreso tra 1 e 10 (1 massima e 10 minima).
- **ID Design Goal**, un identificatore univoco e un nome esplicativo.
- **Descrizione,** una descrizione del design goal.
- **Categoria**, ovvero la categoria di appartenenza del design goal.

- **RNF di origine**, ovvero il requisito non funzionale che lo ha generato.

DESIGN GOALS

Rank	ID Design Goal	Descrizione	Categoria	RNF di origine
5	DG_1 Sicurezza dei dati	Il sistema deve garantire la sicurezza dei dati sensibili di ogni cliente, utilizzando protocolli di comunicazione sicuri e crittografando le password.	Dependability	RNF_A_2
6	DG_2 Gestione dei permessi	Il sistema deve garantire che gli accessi siano separati per ruolo utente, in modo da consentire solo le operazioni autorizzate per ogni ruolo.	Dependability	RNF_A_3
4	DG_3 Robustezza	Il sistema deve intercettare e rispondere con messaggi di errore ad input errati.	Dependability	RNF_A_4
7	DG_4 Disponibilità	Il sistema dovrà essere disponibile 24/24h e 7/7gg, eccetto nei periodi di manutenzione programmata che devono essere attuati tra le 23:00 e le 5:00 del sabato.	Dependability	RNF_P_2
1	DG_5 Conferma ordine	Il sistema deve mostrare una schermata di conferma dopo che	End user	RNF_U_3

		l'utente ha effettuato l'ordine.		
3	DG_6 Interfaccia web responsive	Il sistema deve essere provvisto di interfaccia responsive	End user	RNF_U_2
2	DG_7 Input errati	Il sistema deve, in caso di input errato da parte dell'utente durante la compilazione di un form, evidenziare i campi scorretti e, se necessario, suggerire la corretta compilazione.	End user	RNF_U_1
9	DG_8 Tempi di risposta	Il sistema deve garantire un tempo di risposta non superiore a 5 secondi	Performance	RNF_P_3
10	DG_9 Navigazione concorrente	Il sistema dovrà essere correttamente funzionante anche con un elevato (100) numero di utenti connessi in contemporanea.	Performance	RNF_P_1
8	DG_10 Modularità	L'applicazione web deve avere un'architettura modulare in modo da facilitare la manutenzione ed evoluzione del sistema.	Mantenance	RNF_S_2

Trade-off

TRADE-OFF	DESCRIZIONE
Tempo di rilascio vs. funzionalità	Se i tempi di rilascio sono stringenti, possono essere rilasciate meno funzionalità di quelle richieste, ma nei tempi giusti.
Tempo di rilascio vs. qualità	Se i tempi di rilascio sono stringenti, si può scegliere di lanciare il prodotto in tempo sacrificando parzialmente la qualità. Ciò può comportare la presenza di bug minori, prestazioni non ottimali o una stabilità non perfetta, accettando questi compromessi per rispettare le scadenze.
Uso della Memoria vs. Velocità	Il sistema migliora le prestazioni mantenendo in sessione dati come reparti, carrello, wishlist e informazioni utente, evitando query al database durante il runtime. Questo aumenta la velocità evitando query al database. Tuttavia, questo incrementa l'uso di memoria sul server.

1.3 Definizioni, acronimi e abbreviazioni

Vengono riportati di seguito alcune definizioni presenti nel documento corrente:

- **Sottosistema:** un sottoinsieme dei servizi del dominio applicativo, formato da servizi legati da una relazione funzionale.
- **Design Goal:** le qualità sulle quali il sistema deve essere focalizzato.
- **Dati Persistenti**: dati che sopravvivono all'esecuzione del programma che li ha creati e che dunque vengono salvati.
- **Mapping Hardware/Software**: studio della connessione tra parti fisiche e logiche di cui si compongono il sistema.
- **SDD:** System Design Document
- **RAD:** Requirements Analysis Document
- **MySQL**: È un database Open Source basato sul linguaggio SQL, composto da un client a riga di comando e un server.
- **DBMS**: Database Management System.

Subsystem decomposition: Scomposizione in sottosistemi e responsabilità.

- **Hardware/software mapping**: Assegnazione hardware e gestione dei componenti.
- **Persistent data management**: Gestione e infrastruttura dei dati persistenti.
- **Access control and security**: Modello di controllo degli accessi e problemi di sicurezza.
- **Global software control**: Meccanismi di controllo globale e sincronizzazione.
- **Boundary conditions**: Comportamento di avvio, arresto e gestione degli errori.

1.4 Organizzazione del documento

Il presente documento di System Design consta di quattro sezioni:

1. **Introduzione**: Fornisce una panoramica generale dell'architettura del software e dei design goal.
2. **Architettura software corrente**: Descrive il sistema esistente (o, se non presente, le architetture di sistemi simili), evidenziando le basi, le ipotesi e i problemi che il nuovo sistema intende affrontare.
3. **Architettura di sistema proposta**: Documenta il nuovo modello di progettazione, suddiviso in:
   - **Overview**: Visione generale del sistema e assegnazione delle funzionalità.
4. **Subsystem services**: Elenca i servizi forniti da ciascun sottosistema, fungendo da riferimento per definire le interfacce e i confini tra i sottosistemi.

2 Architettura del sistema corrente

L'applicazione web "AEMME" è un e-commerce costruito sulla base dell'e-commerce di La Feltrinelli. Quest'ultimo ecosistema ruota attorto alla piattaforma lafeltrinelli.it, che gestisce un catalogo ricco di articoli di vario genere. La Feltrinelli dispone di programmi di fidelizzazione come Feltrinelli Card, di profonda ispirazione per i requisiti dell'e-commerce "AEMME".
Dal punto di vista tecnico, il sistema include:

1. Database centralizzato per la gestione delle informazioni sui clienti e prodotti;
2. Marketing automation: utilizzano big data e comunicazioni personalizzate (e-mail, SMS e notifiche push);
3. Integrazione con sistemi loyalty e di profilazione utenti per esperienze one-to-one.

Questo sistema possiede un'architettura organizzata come:

- Interfaccia utente: LaFeltrinelli.it e app mobile;
- Logica applicativa: gestione utenti, prodotti, ordini, wishlist, loyality;
- Storage: database centralizzato per contenuti e dati utenti.

3 Architettura sistema proposto

3.1 Panoramica sulla sezione

La software architecture che abbiamo scelto è una three-tier.
Questa architettura consente di distinguere:

- l'Interface Layer, che include tutti i boundary object che interagiscono con l'utente;
- l'Application Logic Layer, che comprende tutti gli oggetti relativi al controllo e alle entità responsabili dell'elaborazione, delle regole di verifica e delle notifiche richieste dall'applicazione;
- lo Storage Layer, che si occupa della memorizzazione, del recupero e dell'interrogazione di oggetti persistenti.

La separazione dell'interfaccia dalla logica applicativa consente di modificare o sviluppare diverse interfacce utente mantenendo invariata la logica applicativa.

3.2 Decomposizione in sottosistemi

Gestione ordine: si occupa della gestione dei prodotti nel carrello, della procedura di acquisto, la visualizzazione degli ordini e la modifica dello stato ordine.

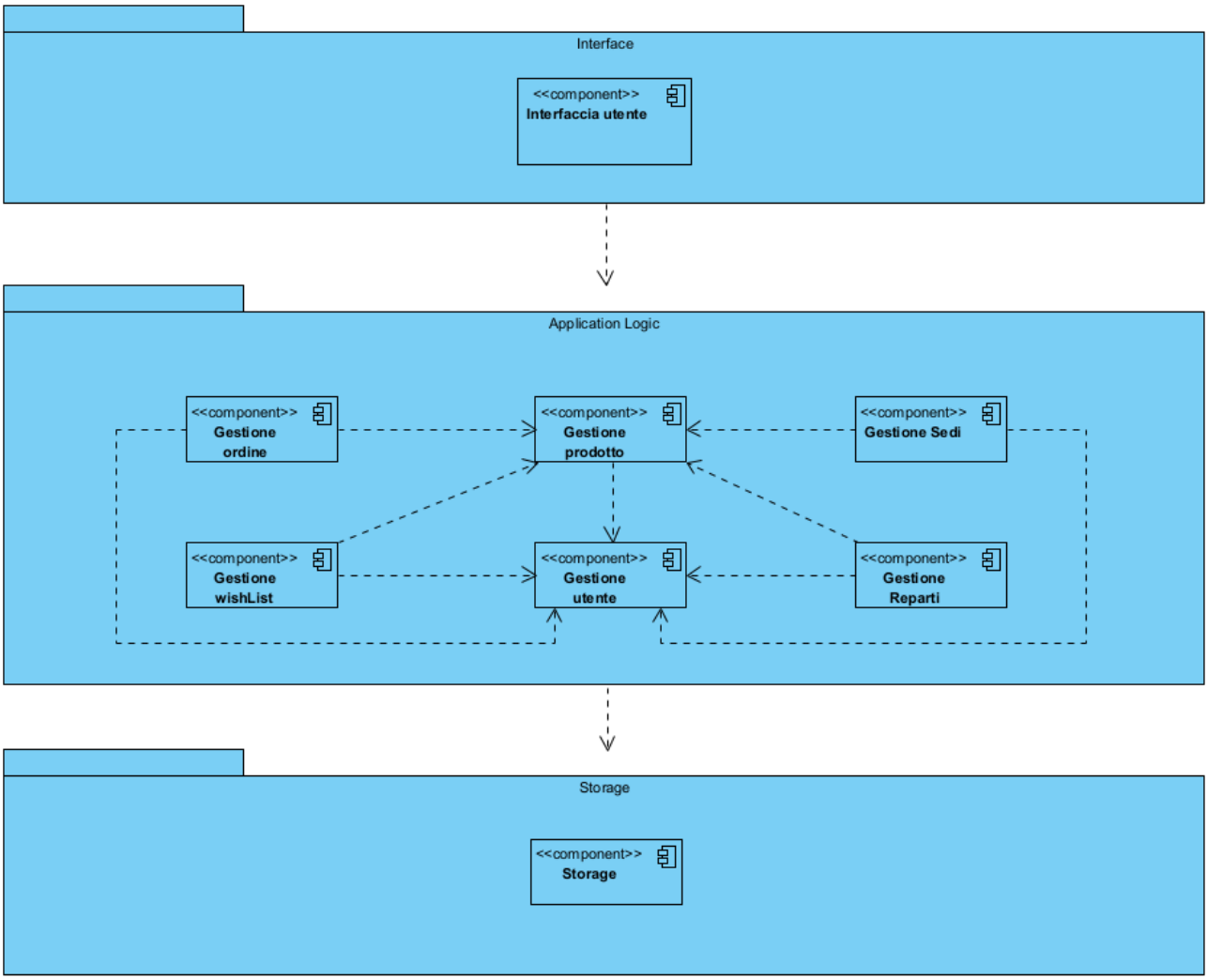
Gestione prodotto: si occupa dell'inserimento, della modifica e la visualizzazione di un libro, e la ricerca nel catalogo.

Gestione sedi: si occupa dell'inserimento e dell'eliminazione delle sedi, dell'aggiunta e della rimozione di libri da esse.

Gestione wishlist: si occupa dell'inserimento e della rimozione dei libri dalla wishlist e della visualizzazione della stessa.

Gestione utente: si occupa dell'autenticazione, della registrazione, della gestione dell'area personale dell'utente registrato e della tessera di ogni cliente premium.

Gestione reparti: si occupa della gestione completa dei reparti e dei libri associati.



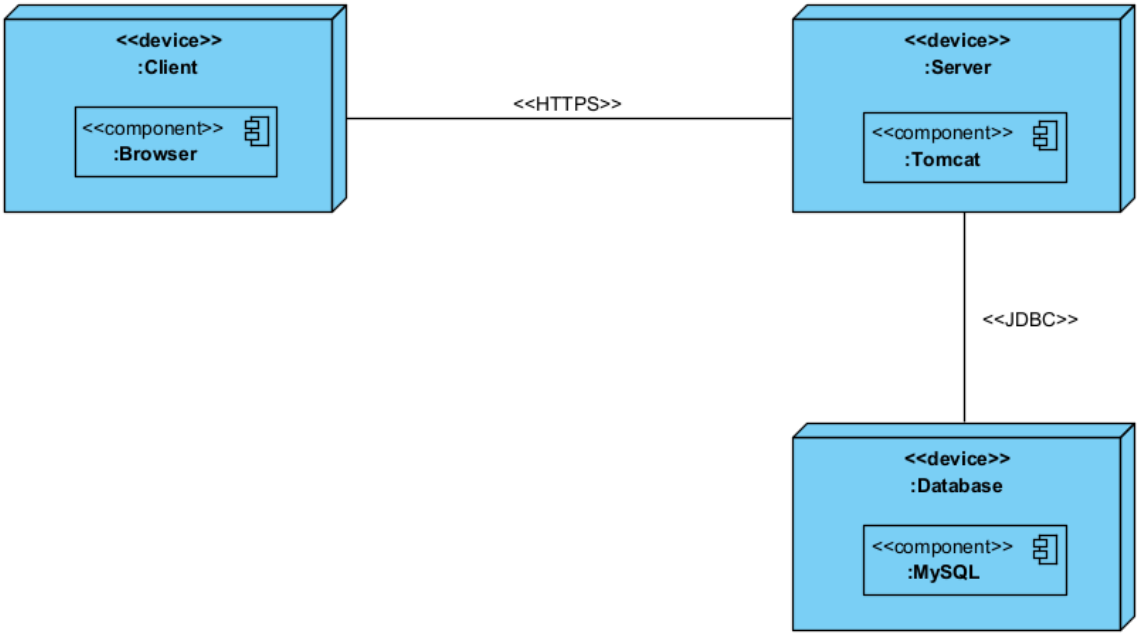
3.3 Mapping hardware/software

Il sistema che si desidera sviluppare utilizzerà una struttura hardware costituita da un Server che risponderà ai servizi richiesti dai client.

Le componenti hardware e software necessarie per il client sono un computer dotato di connessione internet e di un web browser installato su di esso.

Per il server c'è necessità di una macchina con connessione ad Internet e con la capacità di immagazzinare una grande quantità di dati. La componente software necessaria è dunque un DBMS, per consentire la comunicazione con più client.

Il Web Server è rappresentato da Apache Tomcat 9 ed è situato su una singola macchina, la logica del sistema è costituita dalla tecnologia Java Servlet mentre l'interfaccia utente è realizzata utilizzando pagine JSP (Java Servlet Pages). Il client è rappresentato dal Web Browser utilizzato dall'utente. La comunicazione tra i nodi è rappresentata da richieste e risposte HTTP tra client e server, e da query in JDBC tra server e database.



3.4 Gestione dei dati persistenti

Introduzione

Per la gestione del salvataggio dei dati persistenti del sistema si è deciso di utilizzare un database relazionale, al fine di gestire agevolmente l'accesso concorrente ai dati e contemporaneamente garantire la consistenza dei dati tramite l'utilizzo di un DBMS.

La scelta di utilizzo di un DBMS è stata presa al fine di mantenerci quanto più possibile coerenti con i design goals stabiliti, potendo contare su:

- **Imposizioni di vincoli di integrità sui dati,** poiché un DBMS permette di specificare diversi tipi di vincoli per mantenere l'integrità dei dati e controlla che tali vincoli siano soddisfatti quando la base di dati cambia.
- **Privatezza dei dati**, garantita dal fatto che un DBMS permette un accesso protetto ai dati.
- **Affidabilità dei dati**, infatti un DBMS offre dei metodi per salvare copie dei dati e per ripristinare lo stato della base di dati in caso di guasti software e hardware.

I dati che devono essere resi persistenti sono:

- **Utente**: l'oggetto che rappresenta l'utente registrato con i suoi dati personali;
- **Libro**: l'oggetto che rappresenta il prodotto offerto dall'e-commerce con tutte le informazioni ad esso associate;

- **Carrello**: utile a mantenere la lista dei libri che un cliente è intenzionato ad acquistare;
- **WishList**: utile a mantenere la lista dei libri che un cliente registrato preferisce;
- **Reparto**: utile a categorizzare i libri del catalogo;
- **Ordine**: utile a mantenere la lista dei libri acquistati e tutte le informazioni necessarie (es: dataEffettuazioneOrdine);
- **Sede**: rappresenta la sede fisica dell'e-commerce AEMME;
- **Tessera**: utile a mantenere i punti fruibili dai clienti premium per gli acquisti.

3.5 Controllo degli accessi e sicurezza

Attori / Oggetti	Ospite	Cliente Standard	Cliente Premium
Gestione Utente	registrazioneCliente;	login, logout, visualizzaAreaPersonale, modificaNomeUtente, modificaPassword, modificaNumeriDiTelefono, diventaPremium;	login, logout, visualizzaAreaPersonale, modificaNomeUtente, modificaPassword, modificaNumeriDiTelefono, diventaStandard, visualizzaStatoTessera;
Gestione Ordini	inserisciLibroNelCarrello, rimuoviLibroDalCarrello, visualizzaCarrello, modificaQuantita;	inserisciLibroNelCarrello, rimuoviLibroDalCarrello, visualizzaCarrello, modificaQuantita, acquistaOra, modificaQuantitaRevisione, procediOrdine, visualizzaStoricoOrdini, visualizzaDettagliOrdine;	inserisciLibroNelCarrello, rimuoviLibroDalCarrello, visualizzaCarrello, modificaQuantita, acquistaOra, modificaQuantitaRevisione, procediOrdine, visualizzaStoricoOrdini, visualizzaDettagliOrdine;
Gestione Prodotto	ricercaLibri, visualizzaDettagliLibro;	ricercaLibri, visualizzaDettagliLibro;	ricercaLibri, visualizzaDettagliLibro;
Gestione Reparto	visualizzaReparto;	visualizzaReparto;	visualizzaReparto;
Gestione Sedi			
Gestione WishList		inserisciLibroNellaWL, rimuoviLibroDallaWL, visualizzaWL;	inserisciLibroNellaWL, rimuoviLibroDallaWL, visualizzaWL;

Attori / Oggetti	Gestore ordini	Gestore sedi	Gestore catalogo
Gestione Utente	login, logout, visualizzaAreaPersonale, modificaPassword, modificaNumeriDiTelefono;	login, logout, visualizzaAreaPersonale, modificaPassword, modificaNumeriDiTelefono;	login, logout, visualizzaAreaPersonale, modificaPassword, modificaNumeriDiTelefono;
Gestione Ordini	modificaStatoOrdine, modificaCorriere, visualizzaOrdini, visualizzaDettagliOrdine;		
Gestione Prodotto			inserisciNuovoLibro, rimuoviLibro, modicaDatiLibro, addToStore, ritira, visualizzaCatalogo;
Gestione Reparto			visualizzaLibriNelReparto, inserisciLibroNelReparto, rimuoviLibroDalReparto, modificaDatiReparto, visualizzaReparti, rimuoviReparto, aggiungiReparto;
Gestione Sedi		inserisciLibroNellaSede, rimuoviLibroDallaSede, aggiungiSede, rimuoviSede, visualizzaSedi;	
Gestione WishList			

3.6 Controllo globale del software

Il sistema AEMME è un'applicazione interattiva in cui ogni funzionalità viene avviata in seguito a una richiesta generata dall'utente tramite un'interfaccia grafica. Quando l'utente desidera accedere a una funzionalità, invia un comando che viene interpretato dal sistema.

La richiesta viene gestita da un componente dedicato che analizza l'input, esegue le operazioni necessarie e produce una risposta. Questo componente

indirizza il flusso verso i moduli responsabili della logica di controllo e della logica applicativa, garantendo una corretta elaborazione.

Essendo una web-application, il sistema adotta un approccio basato su eventi (*event-driven*), dove ogni azione dell'utente genera un evento che viene elaborato secondo il flusso previsto dal sistema.

3.7 Condizioni limite

Nel nostro progetto non è previsto lo sviluppo di software per la gestione delle condizioni limite, poiché questa è delegata automaticamente a Tomcat e MySQL.

4 Servizi dei sottosistemi

Sottosistema Gestione Utente

Servizio	Descrizione	Operazioni
Registrazione cliente	Questo servizio permette a un cliente di registrarsi all' e-commerce.	registrazioneCliente;
Autenticazione	Questo servizio permette di effettuare l'accesso al sistema tramite le proprie credenziali per sfruttare tutte le funzionalità che offre e disconnettersi dal sistema.	login; logout;
Gestione Area utente	Questo servizio permette di visualizzare i dati relativi alla propria area utente, modificare i dati relativi alla propria area utente.	modificaNomeUtente; modificaPassword; modificaNumeriDiTelefono; diventaStandard; diventaPremium; visualizzaStatoTessera; visualizzaAreaPersonale;

Sottosistema Gestione Ordini

Servizio	Descrizione	Operazioni
Gestione carrello	Questo servizio permette di inserire e rimuovere libri dal carrello, visualizzare il proprio carrello e modificarne la quantità.	inserisciLibroNelCarrello; rimuoviLibroDalCarrello; visualizzaCarrello; modificaQuantita;

Gestione ordine	Questo servizio permette di effettuare la procedura di acquisto dei libri disponibili nel carrello, permette al cliente di visualizzare lo storico degli ordini che ha effettuato.	acquistaOra; modificaQuantitaRevisione; procediOrdine; visualizzaStoricoOrdini; visualizzaDettagliOrdine;
Controllo ordini	Questo servizio permette al gestore degli ordini di gestire lo stato degli ordini e modificare il corriere.	modificaStatoOrdine; modificaCorriere; visualizzaOrdini; visualizzaDettagliOrdine;

Sottosistema Gestione Prodotto

Servizio	Descrizione	Operazioni
Gestione prodotti	Questo servizio permette di aggiungere e rimuovere libri dal catalogo, modificare i dati relativi a un libro e visualizzare i libri del catalogo.	inserisciNuovoLibro; rimuoviLibro; modicaDatiLibro; addToStore; ritira; visualizzaCatalogo;
Gestione catalogo	Questo servizio permette di visualizzare i dettagli del libro e cercare i libri presenti nel catalogo.	ricercaLibri; visualizzaDettagliLibro;

Sottosistema Gestione Reparto

Servizio	Descrizione	Operazioni
Gestione Reparto	Questo servizio permette di aggiungere e rimuovere libri da un reparto e modificare i dati del reparto e aggiungere e rimuovere reparti.	visualizzaLibriNelReparto; inserisciLibroNelReparto; rimuoviLibroDalReparto; modificaDatiReparto; visualizzaReparti; rimuoviReparto; aggiungiReparto;
Visualizza Reparto	Questo sistema permette di visualizzare un reparto.	visualizzaReparto;

Sottosistema Gestione Sedi

Servizio	Descrizione	Operazioni
Gestione Sede	Questo servizio permette di aggiungere e rimuovere libri da una sede e aggiungere e rimuovere sedi.	inserisciLibroNellaSede; rimuoviLibroDallaSede; aggiungiSede; rimuoviSede; visualizzaSedi;

Sottosistema Gestione WishList

Servizio	Descrizione	Operazioni
Gestione WishList	Questo servizio permette di visualizzare la propria WishList, aggiungere e rimuovere libri dalla WishList	inserisciLibroNellaWL; rimuoviLibroDallaWL; visualizzaWL;

ODD

Object Design Document

AEMME

Design Patterns

Nella presente sezione si andranno a descrivere e dettagliare i design patterns da utilizzare nello sviluppo dell'applicativo AEMME.
Per ogni pattern si darà:

- Una brevissima introduzione teorica.
- Il problema che doveva risolvere all'interno di AEMME.
- Una brevissima spiegazione di come si è risolto il problema in AEMME.

Singleton e Observer

Nel contesto dell'ingegneria del software, i design pattern rappresentano soluzioni architetturali consolidate a problematiche ricorrenti nella progettazione di sistemi software complessi. Il presente documento esamina l'implementazione dei pattern Singleton e Observer nel contesto dell'applicativo AEMME.

Il pattern Singleton si configura come una soluzione architetturale appartenente alla categoria dei pattern creazionali.
La sua principale caratteristica risiede nella capacità di garantire l'unicità dell'istanza di una determinata classe all'interno del sistema, fornendo contemporaneamente un punto di accesso globale e controllato a tale istanza. Questo approccio risulta particolarmente efficace in scenari che necessitano di una gestione centralizzata e atomica delle risorse di sistema.

Il pattern Observer si colloca nella categoria dei pattern comportamentali e definisce una relazione uno-a-molti tra oggetti, tale che al variare dello stato di un oggetto (denominato Soggetto), tutti gli oggetti dipendenti (Osservatori) vengano automaticamente notificati e aggiornati. Questa architettura promuove un basso accoppiamento tra le componenti, facilitando la manutenibilità e l'estensibilità del sistema.

Implementazione del Pattern Singleton

La gestione delle connessioni al database è stata implementata attraverso la classe ConPool, che incapsula la logica del pattern Singleton. Tale implementazione si caratterizza per:

- Garanzia di unicità dell'istanza attraverso costruttore privato
- Metodo statico getConnection() per l'accesso controllato all'istanza
- Ottimizzazione delle prestazioni mediante lazy initialization

Observer per notifiche e-mail (implementazione futura):

In ottica evolutiva, il pattern Observer sarà utilizzato per realizzare un sistema di notifiche e-mail. La progettazione prevede:

- Soggetto: La classe che rappresenta il catalogo dei prodotti agirà come soggetto, generando notifiche ogni volta che si verificano eventi rilevanti (ad esempio, l'arrivo di nuovi prodotti o il ripristino della disponibilità di articoli esauriti).
- Osservatori: Gli utenti registrati potranno iscriversi come osservatori, ricevendo notifiche personalizzate in base alle loro preferenze e interazioni con la piattaforma.

Questo approccio consentirà una comunicazione efficiente tra il sistema e gli utenti, migliorando la loro esperienza e incentivando il coinvolgimento con la piattaforma.

L'adozione dei pattern Singleton e Observer consentirà di realizzare un'architettura robusta e scalabile, caratterizzata da:

- Gestione efficiente delle risorse di sistema
- Elevata coesione e basso accoppiamento tra componenti
- Facilità di manutenzione ed estensione

Tale approccio architetturale si dimostra particolarmente efficace nel contesto di un sistema e-commerce, dove la gestione ottimizzata delle risorse e la comunicazione efficiente tra componenti rappresentano requisiti fondamentali per il successo dell'applicazione.