Università degli Studi di Salerno

Corso di Ingegneria del Software

EdenJewelry
SDD_EdenJewelry
Versione 2.3

Data: 03/01/2025

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Alessandro Di Palma	0512115939
Gaetano D'Alessio	0512110836
Luigi Montuori	0512117799
Miriam Eva De Santis	0512117121

Scritto da:	Luigi Montuori

Revision History

Data	Versione	Descrizione	Autore
17/11/2024	1.0	Prima stesura del file di System Design	Luigi Montuori
18/11/2024	1.1	Continuo criteri di affidabilità	Luigi Montuori
20/11/2024	1.2	Scritti capitoli 1 e 2	Gaetano D'Alessio
21/11/2024	1.3	Scritti 3.4, 3.5 e 3.6	Luigi Montuori
21/11/2024	1.4	Fine stesura 3.1	Gaetano D'Alessio
21/11/2024	1.5	Aggiunta Deployment Diagram	Luigi Montuori
21/11/2024	1.6	Mapping hardware/software	Miriam Eva De Santis
23/11/2024	1.7	Lavori ultimati sul file	Gaetano D'Alessio
23/11/2024	1.8	Correzione dell'indice	Alessandro Di Palma
21/12/2024	1.9	Rimozione Boundary Condition	Luigi Montuori
22/12/2024	2.0	Modifiche decomposizione in sottosistemi	Luigi Montuori
03/01/2025	2.1	Aggiunta servizi offerti	Luigi Montuori
03/01/2025	2.2	Correzione par. 3.3 e 3.4	Miriam Eva De Santis
03/01/2025	2.3	Ultime modifiche	Gaetano D'Alessio

Indice

1. INTRODUZIONE

1.1. Scopo del sistema

L'e-shop *EdenJewelry* si propone di offrire una soluzione che venga incontro alle esigenze dei clienti e dei venditori.
I primi cercano un catalogo ampio da consultare e vogliono la comodità degli shop online, per comprare i loro gioielli preferiti. I secondi, necessitano di una piattaforma per vendere i propri prodotti e interfacciarsi con i clienti.
I progettisti, quindi, si trovano a dover effettuare una scelta tra le duplici esigenze degli attori.
Alla fine, si è scelto di dare priorità alle richieste dei venditori. Lo scopo del sistema è allora quello di fornire un e-shop di gioielli con tutte le funzionalità che essi ritengono rilevanti.

1.2. Obiettivi di progettazione

Definizione delle priorità

1. Priorità alta: funzionalità fondamentale, che necessita di essere implementata fin dalla prima release.
2. Priorità media: funzionalità meno rilevante, che può essere implementata nelle successive release.
3. Priorità bassa: funzionalità facoltativa, non necessita di esser implementata.

1.2.1. Criteri di prestazione

1. **Memoria**
   Tutti i dati riguardo gli utenti, i prodotti e gli ordini devono essere inseriti e conservati in un database relazionale.
   Priorità: Alta.
2. **Concorrenza**
   Il sistema deve gestire diversi accessi in contemporanea.
   Priorità: Alta
3. **Tempo di risposta**
   I tempi di attesa devo essere ridotti il più possibile.
   Priorità: Bassa.
4. **Throughput**
   Il throughput deve essere abbastanza alto per soddisfare un carico sempre crescente.
   Priorità: Alta.

1.2.2. Criteri di affidabilità

1. **Sicurezza**
   Deve garantire l'accesso alle varie funzionalità soltanto agli utenti autorizzati (le funzionalità del venditore non devono essere accessibili da utenti normali)
   Priorità: Alta.
2. **Attendibilità**
   Il sistema deve essere in grado di garantire l'attendibilità dei propri servizi.

(Quando va a buon fine un acquisto, oltre a garantire la disponibilità del prodotto va anche aggiornato il dato per gli altri utenti)
Priorità: Media.

1.2.3. Criteri di costo

1. **Stock del catalogo**
   E da prevedere un costo (hours/men) per inserire i prodotti esistenti nel catalogo.
   Priorità: Alta.
2. **Manutenzione**
   Ci saranno degli sviluppatori addetti alla manutenzione del sito.
   Priorità: Bassa.
3. **Sviluppo**
   Il costo dello sviluppo, finita la fase di progettazione, sarà congruo alle funzionalità offerte.
   Priorità: Alta.

1.2.4. Criteri di mantenimento

1. **Portabilità**
   È possibile fare il deploy del sistema sviluppato su qualunque macchina provvista del container "Apache Tomcat".
   Priorità: Alta.

1.2.5. Criteri di End User

1. **Responsività**
   Il sistema deve essere visualizzabile su dispositivi differenti.
   Priorità: Media.

1.2.6 Trade-off tra obiettivi

Funzionalità Vs Usabilità	Abbiamo preferito creare uno shop semplice da usare, piuttosto che concentrarci su implementare molte funzionalità ridondanti.
Costo Vs Robustezza	Un giusto compromesso tra un sito robusto, ma che non richieda troppo tempo per essere implementato.
Sviluppo rapido Vs Funzionalità	Favorire e facilitare lo sviluppo, riducendocelo ad una serie di funzionalità essenziali.

2. Architetture software simili

Nel nostro caso non abbiamo un'architettura software esistente sulla quale basarci. Abbiamo quindi deciso di confrontarci con gli e-shop di gioielli attualmente presenti sul mercato.
I siti che sono stati valutati sono:

1. **Pandora**, marchio leader nel campo della bigiotteria e gioielli, anche di lusso;
2. **Marlù**, competitor che è riuscito a ricavarsi la sua nicchia di utenti negli anni.

Entrambi suddividono i prodotti in categorie, hanno un'area utente dedicata e un carrello consultabile prima del check-out. Inoltre, troviamo anche una wishlist dove è possibile salvare i gioielli desiderati, in attesa di sconti o solo come promemoria. L'area utente è accessibile tramite e-mail e password, proteggendo dati sensibili dai malintenzionati.
Le tecnologie utilizzate sono eterogenee ma sicuramente a gestire la persistenza troviamo un database relazionale.
Queste sono tutte caratteristiche che consideriamo essenziali e abbiamo deciso di riportare nel nostro sistema.

3. Architettura del sistema proposto

3.1 Decomposizione in sottosistemi

La scelta per la decomposizione in sottosistemi è un'architettura a Tre Layers.
I sottosistemi sono i seguenti:

- **Gestione Account**: rappresenta la gestione degli utenti, con tutte le funzionalità che lo riguardano;
- **Gestione Prodotti**: si occupa della presentazione del catalogo e dei prodotti che lo compongono, della loro descrizione e così via;
- **Gestione Ordini**: gestisce gli ordini dei vari utenti, il loro storico e i dettagli dell'acquisto.

La scelta di questo pattern architetturale non è lasciata al caso. Non solo quest'ultimo è particolarmente adatto allo sviluppo di un sito di e-commerce, ma la decomposizione così effettuata tende a diminuire la complessità del sistema.
Quello che vogliamo ottenere è la massima coesione e il minimo accoppiamento.
La coesione misura la dipendenza tra le classi, mentre l'accoppiamento la dipendenza tra i sottosistemi.
Nello stile architetturale scelto la coesione è alta e i sottosistemi comunicano attraverso le loro interfacce.
La macchina virtuale, tra l'altro è aperta (trasparente). Questa permette al layer più alto (*upper layer*) di utilizzare i servizi del layer più basso (*bottom layer*), garantendo prestazioni migliori.

Conseguentemente, illustriamo i componenti appartenenti ai tre layers sopracitati.
La **Gestione Account** è costituita dalle seguenti componenti:

- AreaUtente si occupa di far visualizzare l'area riservata dell'utente;
- AreaVenditore si occupa di far visualizzare l'area riservata ai venditori;
- Autenticazione, rappresenta l'area dove il venditore può effettuare il login ed il logout.

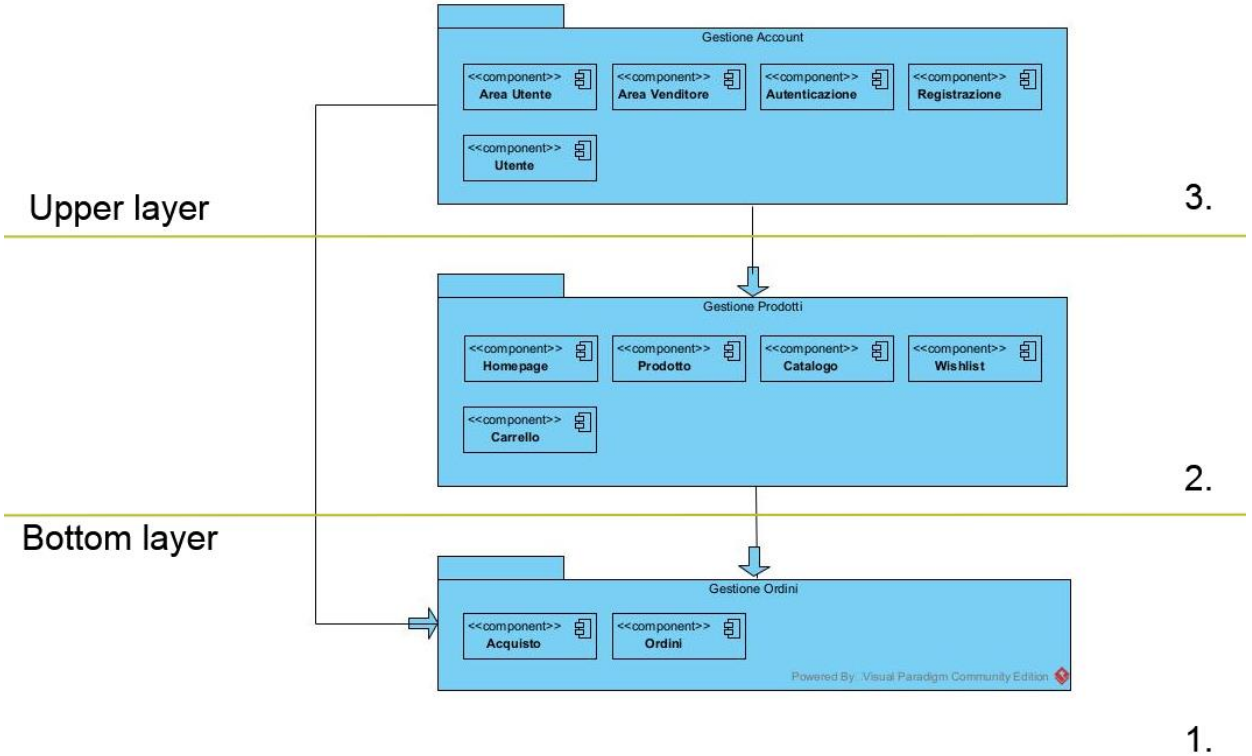
- Registrazione, rappresenta l'area dove gli utenti non registrati in precedenza, possono farlo.
- Utente, contiene i dati relativi ad utente e venditori (tutti quelli registrati al sito).

Il layer **Gestione Prodotti** è composto dalle componenti:

- HomePage, l'area "principale" del sito, visualizzata appena si accede.
- Prodotto, l'insieme di tutti i servizi inerenti ai gioielli offerti, la loro aggiunta, la loro rimozione, modifica e così via.
- Catalogo, si interpone per gestire gli oggetti catalogo.
- Wishlist, dove (come si può intuire dal nome) l'utente può aggiungere i gioielli che desidera.
- Carrello, che si occupa di raggruppare i vari gioielli che l'utente è interessato ad acquistare.

Il layer **Gestione Ordini** è composto, infine, dalle componenti:

- Acquisto, area dove l'utente inserisce tutti i dati riguardo il proprio acquisto.
- Ordini, grazie al quale si possono visualizzare lo storico degli acquisti passati e i dettagli dei singoli ordini,



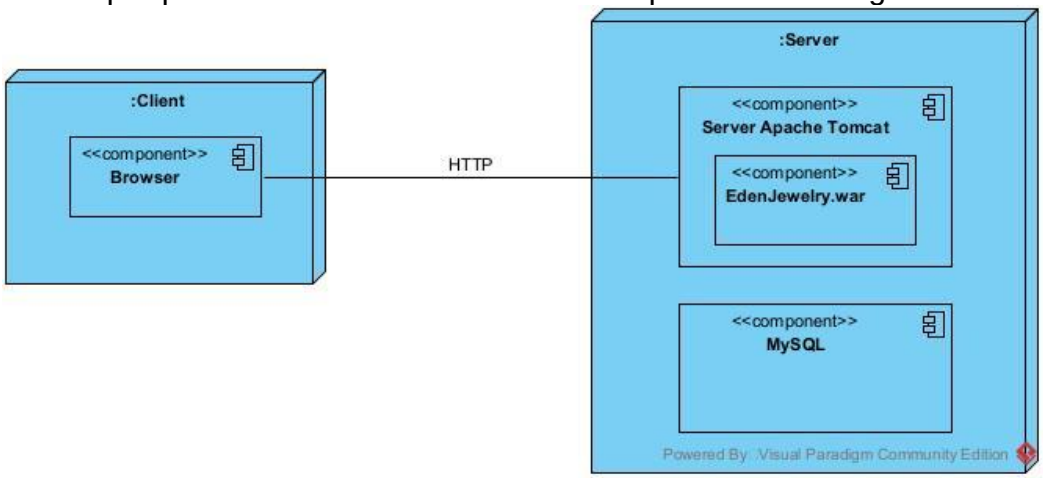
3.2. Mapping Hardware/Software

Per gestire i vari aspetti del nostro shop (frontend, backend, persistenza), abbiamo deciso di affidarci ad una serie di tecnologie eterogenee.
Il sito è accessibile online tramite una connessione http al server, ossia la macchina fisica dove è stato installato e gira il sistema. Per poter fruire del sistema, il client interagisce con la parte frontend tramite il browser. Per gestire la comunicazione tra

client e server si utilizzano request/ response HTTP. Il deployment del sistema viene effettuato sul container Apache TomCat del host.

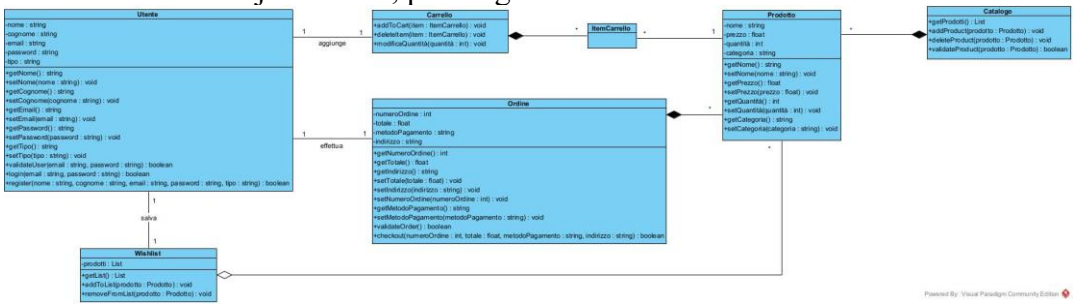
-Componenti hardware:
• Server (fisicamente una macchina x86):

-Componenti software:
• Apache Tomcat: verrà utilizzato per la risoluzione del codice del backend e la gestione di request/response;
• MySQL: per l'archiviazione di dati, principalmente persistenti (cioè necessari per un buon funzionamento del sito). Il database viene ricavato dal modello EER;
• JavaScript: sfruttati per dare un contesto più interattivo e personalizzato all'utente specifico;
• Jsp: permettono di aggiungere codice Java all'interno di file HTML per la generazione di contenuti dinamici;
• Servlet: oggetto Java, residente lato server, che comunica con il client tramite request e response HTTP.
• HTML: per lo "scheletro" del front end, che andremo poi a modificare con le JSP.
• CSS: per personalizzare lo stile del sito e la parte front-end generale



3.3. Gestione della persistenza

Come già detto poc'anzi, per la gestione della persistenza utilizzeremo un database relazionale. La scelta del DBMS è ricaduta su MySQL. Di seguito è presentata una raffinazione dell'object model, per meglio identificare le relazioni tra le classi/entità.



Da questa rappresentazione ricaviamo la tabella dei dati persistenti, con i relativi

attributi.

Utente:

- nome
- cognome
- email
- password
- tipo

Prodotto:

- nome
- prezzo
- quantità
- categoria

Ordine:

- numeroOrdine
- totale
- metodoPagamento
- indirizzo

Wishlist:

- prodotti

3.4. Controllo degli accessi

		ATTORI		
		Utente	Utente non registrato	Venditore
OGGETTI	Info Utente	Login() Logout()	Register()	Login() Logout()
	Prodotti	getProdotti() cercaProdotto()	getProdotti() cercaProdotto()	getProdotti () cercaProdotto()
	Catalogo	getProdotti ()	getProdotti()	getProdotti() addProduct() deleteProduct()
	Carrello	visualizzaCarrello() addToCart() deleteItem() modificaQuantita()		visualizzaCarrello() addToCart() deleteItem() modificaQuantita()
	Ordine	visualizzaOrdini() getNumeroOrdine()		visualizzaOrdini() getNumeroOrdine()

3.5. Controllo globale del Software

Essendo un e-commerce, il controllo globale del software è centralizzato, per la

precisione *event-driven*, il dispatcher si occuperà dello smistamento delle richieste http verso le varie Servlet, che a loro volta gestiranno la richiesta.

4. Subsystem services

Infine, illustriamo i servizi esposti dai sottosistemi.

Servizi Offerti

Gestione Account

Registrazione	Register	Consente agli utenti non registrati di effettuare la registrazione.
Autenticazione	Login	Consente agli utenti di effettuare il login una volta inserite le proprie credenziali.
	Logout	Consente agli utenti loggati di uscire dal proprio account.

Gestione Prodotti

Homepage	prodottiHomepage	Consente ai venditori di scegliere quali prodotti mostrare nella pagina principale, che ovviamente verranno visualizzati dall'utente.
CatalogoVenditore	visualizzaProdotto	Consente al venditore di visualizzare i dettagli di un prodotto.
	aggiungiProdotto	Consente ai venditori di aggiungere nuovi prodotti al sito.
	modificaProdotto	Consente ai venditori di modificare i dettagli di un prodotto.
	rimuoviProdotto	Consente ai venditori di rimuovere dei prodotti dal sito.
CatalogoUtente	visualizzaProdotto	Consente all'utente di visualizzare i dettagli di un prodotto.
Prodotto	ricercaProdotto	Consente all'utente di cercare un prodotto specifico presente nel

		catalogo.
Wishlist	aggiungiWishlist	Consente all’utente di aggiungere un prodotto alla propria wishlist.
	rimuoveDaWishlist	Consente all’utente di rimuovere un prodotto dalla propria wishlist.
Carrello	addToCart	Consente all’utente di aggiungere un prodotto al carrello.
	removeFromCart	Consente all’utente di rimuovere un prodotto dal carrello.

Gestione Ordini

Acquisto	effettuaAcquisto	Consente all’utente di effettuare l’acquisto.
OrdiniUtente	storicoOrdini	Consente all’utente di visualizzare i propri ordini passati.
OrdiniVenditore	storicoVendite	Consente al venditore di visualizzare tutte le vendite