Università degli Studi di Salerno
Corso di Ingegneria del Software

HomeDecore
System Design Document
Versione 1.5

Data: 26/11/2024

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Alfieri Riccardo	0512116533
Cammarota Lucageneroso	0512116941
Fasolino Pietro	0512116473
Marino Michele Graziano	0512117109

Scritto da:	Lucageneroso Cammarota

Revision History

Data	Versione	Descrizione	Autore
17/11/2024	1.0	Decomposizione del sistema in sottosistemi indipendenti altamente coesi e debolmente accoppiati. Definizione dei design goals.	Full team
21/11/2024	1.1	Component diagram, deployment diagram.	Lucageneroso Cammarota
22/11/2024	1.2	Design goals	Marino Michele Graziano
22/11/2024	1.3	Tavola degli accessi e deployment diagram	Alfieri Riccardo
23/11/2024	1.4	Boundary Conditions	Fasolino Pietro
26/11/2024	1.5	• Revisione del diagramma architetturale • Revisione del control flow • Revisione gestione dati persistenti	Lucageneroso Cammarota
29/11/2024	1.6	Rivisitazione gestione persistenza	Lucageneroso Cammarota

Indice

System Design Document

Sommario

1. Introduzione 4
   - 1.1. Scopo del sistema 4
   - 1.2. Design goals 4
   - 1.2.1 Criteri di usabilità 4
   - 1.2.2 Criteri di affidabilità 5
   - 1.2.3 Criteri di prestazione 6
   - 1.2.4 Criteri di manutenzione 6
   - 1.3. Definizioni, acronimi e abbreviazioni 8
   - 1.4. Riferimenti 9
   - 1.5. Panoramica 9
2. Architettura software proposta 9
   - 2.1. Panoramica 9
   - 2.2. Decomposizione in sottosistemi 10
   - 2.3. Mappatura hardware/software 12
   - 2.4. Gestione dei dati persistenti 13
   - 2.5. Controllo degli accessi e sicurezza 13

1. Introduzione

1.1. Scopo del sistema

L’obiettivo è quello di definire una decomposizione del sistema in sottosistemi , ognuno dei quali dovrà essere più indipendente possibile dall’implementazione degli altri sottosistemi facendo sì che la comunicazione tra essi avvenga sfruttando il passaggio per riferimento tipico dei linguaggi object-oriented, garantendo un basso grado di accoppiamento, e puntando ad un alto grado di coesione tra le componenti di uno stesso modulo. L’architettura sarà chiusa e ogni livello fornirà i suoi servizi al livello

superiore sfruttando i servizi dei livelli sottostanti. Si intende garantire la corretta gestione dei dati a basso livello, coadiuvata da un’efficiente e coerente logica di business che faccia da tramite nell’interazione sistema-utente assicurando un’esperienza user piacevole.

1.2. Design goals

1.2.1 Criteri di usabilità

- **Guida visiva**: Ogni schermata deve fornire indicazioni e messaggi contestuali per guidare l'utente nelle operazioni (es. completamento di un ordine).
  Priorità: Alta.

- **Navigazione strutturata**: Il menu di navigazione deve essere accessibile in ogni pagina e permettere il ritorno rapido alla homepage o al carrello.
  Priorità: Alta.

- **Accesso rapido**: Gli utenti registrati devono poter accedere direttamente alle sezioni chiave (catalogo, carrello) tramite scorciatoie visibili nella dashboard personale.
  Priorità: Alta.

- **Responsive**: L’applicazione web deve essere responsive per permettere agli utenti di utilizzarla da dispositivi di diverse dimensioni.

  Priorità: Alta

1.2.2 Criteri di affidabilità

- **Disponibilità**: Il sistema deve essere disponibile per almeno il 99.9% del tempo durante l'anno.

  Priorità: Bassa.

- **Robustezza:** Tutti gli input devono essere validati lato server e lato client, con messaggi di errore specifici per ogni problema (es. formato errato, campo obbligatorio mancante).

  Priorità: Alta.

- **Sicurezza**: Il sistema deve garantire l'accesso alle proprie risorse, dati sensibili e componenti esclusivamente agli utenti autorizzati.

  Priorità: Media.

- **Tolleranza agli errori**: In caso di guasti, il sistema deve essere in grado di riprendersi automaticamente o notificare l'errore senza perdita di dati importanti.

  Priorità: Bassa.

1.2.3 Criteri di prestazione

- **Tempo di risposta:** Tutte le pagine principali (catalogo, carrello, checkout) devono essere caricate in un massimo di due secondi, dopo l'immissione di una richiesta dell'utente.

  Priorità: Bassa.

- **Throughput**: Il sistema deve essere in grado di elaborare almeno 1000 richieste simultanee in condizioni di carico normale.

  Priorità: Bassa.

- **Scalabilità:** Il sistema deve essere distribuito al fine di poter gestire quantità di richieste variabili a seconda del periodo.

  Priorità: Media.

  Priorità: Bassa.

- Memoria: I dati relativi agli utenti e ai prodotti devono essere resi persistenti in un database relazionale.
  Priorità: Alta

1.2.4 Criteri di manutenzione

- **Modello Three-Tier:** Il sistema deve avere un'architettura a tre livelli: il tier presentazione, il tier applicazione e il tier dati al fine di garantire la manutenibilità e eventuali estensioni del sistema.
  Priorità: Alta.

- **Estensibilità**: Il sistema deve garantire una facile integrazione di nuove funzionalità e/o classi.

Priorità: Alta.

- **Modificabilità**: Il sistema deve garantire la possibilità di modificare facilmente le funzionalità e/o classi già presenti.

  Priorità: Alta.

- **Adattabilità**: Il sistema deve garantire la possibilità di essere facilmente adattato a differenti domini di applicazione.
  Priorità: Bassa.

- **Portabilità**: Il sistema deve garantire la possibilità di essere facilmente adattato a differenti piattaforme.
  Priorità: Bassa.

- **Leggibilità**: Il sistema deve garantire la possibilità di essere facilmente comprensibile leggendo il codice.
  Priorità: Media.

1.3. Definizioni, acronimi e abbreviazioni

- **ORM**: Object-Relational Mapping

JavaEE: Java Platform, Enterprise Edition

- **DAO**: Data Access Object

1.4. Riferimenti

Lo sviluppo del documento si è basato sul modello di analisi proposto nel Requirements Analysis Document (RAD) HomeDecore Versione 1.12

1.5. Panoramica

Il documento descrive l'architettura proposta, la decomposizione in sottosistemi, la mappatura hardware/software, la gestione dei dati persistenti, il controllo degli accessi e la sicurezza, il controllo globale del software e le condizioni limite.

2. Architettura software proposta

2.1. Panoramica

Si intende aderire allo stile architetturale three-tier che definisce un software che sfrutta la View per gestire l'interazione con l'utente, lo strato Control che realizza la logica applicativa e lo strato Model che mantiene la struttura dei dati del dominio.

I servizi e i relativi sottosistemi sono stati individuati grazie ad un iniziale processo di layering che ha definito un'architettura basata su tre livelli:

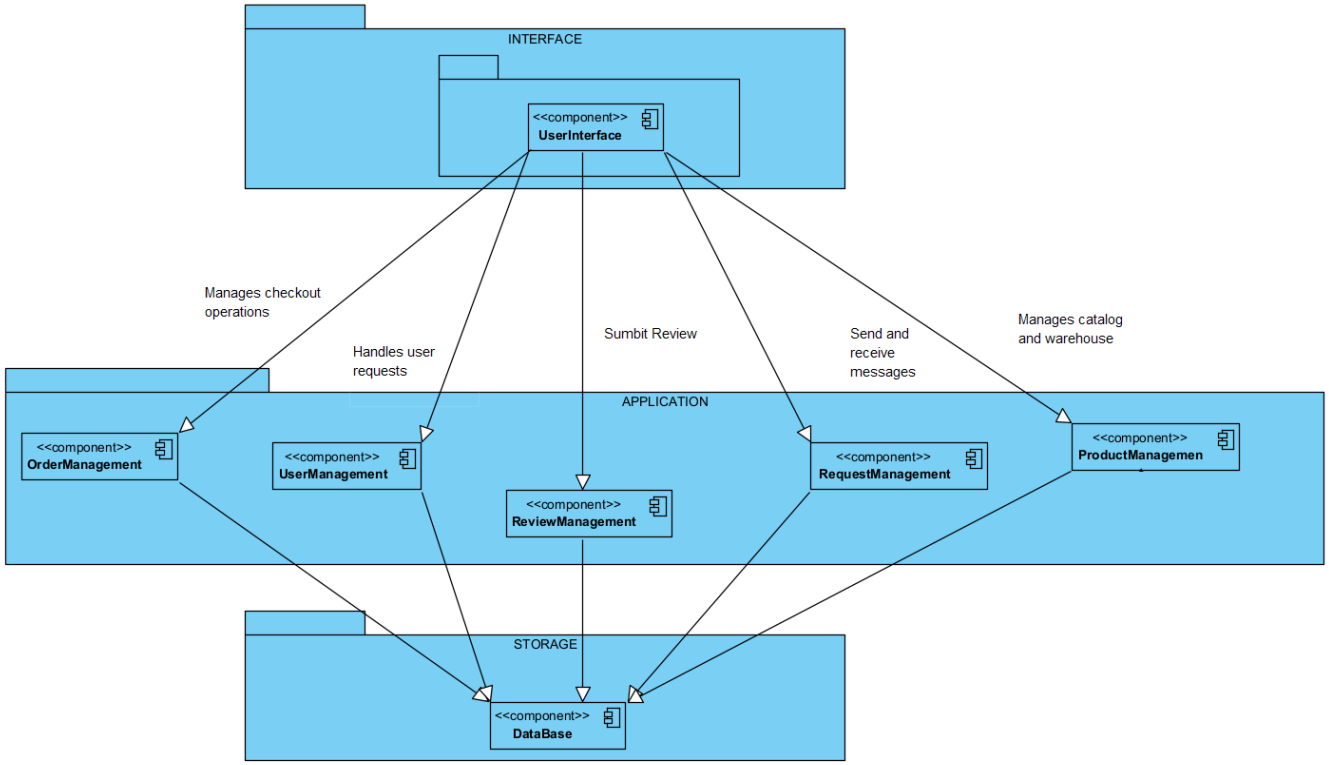
- **Presentation**, che si occupa di gestire l'interazione con l'utente;
- **Business**, che si occupa della logica applicativa;
- **Data Access Layer**, che si occupa della gestione della persistenza dei dati e della manipolazione degli stessi sul database.

Al processo di layering è stato abbinata la tecnica del partitioning, che ha individuato 5 moduli principali:

- la gestione degli utenti;

- la gestione dell'acquisto;
- la gestione del feedback;
- la gestione del catalogo;
- La gestione della richiesta.

2.2. Decomposizione in sottosistemi

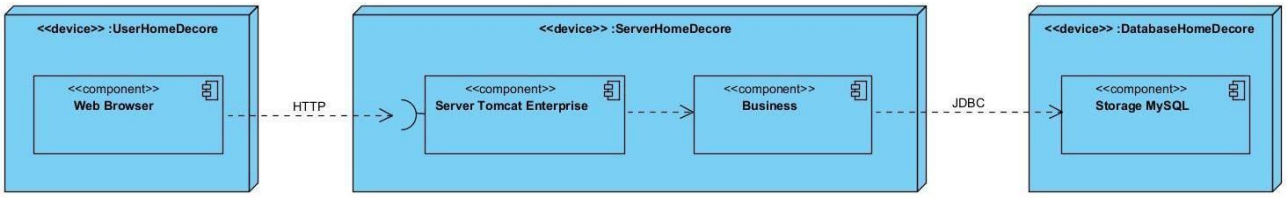


I sottosistemi individuati sono:

- **Gestione Utenti**: Gestisce i processi di registrazione, login, recupero password ecc.
- **Gestione Ordine**: Riceve, elabora, spedisce e fattura gli ordini.
- **Gestione Recensione**: Inserimento, visualizzazione, modifica delle recensioni.
- **Gestione Prodotti**: Gestisce il catalogo dei prodotti, l'inventario, aggiorna le informazioni sui prodotti.
- **Gestione Richiesta**: Permette l'invio e ricezione delle richieste da parte di Magazzinieri, fornitori

e gestori dell’ordine.

2.3. Mappatura hardware/software



Il **Deployment Diagram** mostra la distribuzione fisica dei componenti software su diversi nodi hardware. In questo caso:

- **Client Browser**: Gli utenti accedono tramite un browser, inviando richieste HTTP/HTTPS.

- **Application Server**: Un server TomEE gestisce la logica applicativa e l'orchestrazione dei flussi.
- **Database Server**: MySQL conserva i dati persistenti.

La comunicazione è implementata tramite protocolli noti (HTTP per client-app server, JDBC per app-database).

2.4. Gestione dei dati persistenti

❖ I dati del sistema che si intende rendere persistenti sono:
- Le informazioni degli utenti;
- Le informazioni sui prodotti;
- Le informazioni sugli ordini;
- Le informazioni sulle recensioni;
- Le informazioni sulle richieste.

❖ Per questo fine si utilizzerà un approccio di memorizzazione centralizzato al fine di avere maggior controllo sui dati.

❖ La scelta dello spazio di archiviazione ricade quindi sul database relazionale che risulta essere particolarmente adatto al tipo strutturato dei dati che HomeDecore dovrà gestire.

❖ Inoltre, al fine di garantire l'integrità dei dati, si definiranno dei meccanismi di recupero dati sul DB.

2.5. Controllo degli accessi e sicurezza

Oggetti / Attori	Prodotto	Ordine	Recensione	Carrello	Richiesta
Cliente	Visualizza()	<<create>> Visualizza()	<<create>> Visualizza()	aggiungiProdotto() svuota() visualizza()	
Magazziniere	aggiungiAlMagazzino() modificaProdotto() rimuoviDalMagazzino()	Visualizza()			<<create>>
Gestore Ordini		modificaStato() visualizza()			Accetta()
Fornitore	<<create>> Visualizza()				Accetta()
Guest	Visualizza()		Visualizza()	aggiungiProdotto() svuota() visualizza()	