Università degli Studi di Salerno
Corso di Ingegneria del Software

HomeDecore
System Design Document
Versione 1.5

Data: 26/11/2024

Progetto: HomeDecore	Versione: 1.2
Documento: System Design	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

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 o scrittura di una recensione).

  Priorità: Alta.

- **Navigazione strutturata**: Il menu di navigazione deve essere accessibile in ogni pagina e permettere il ritorno rapido alla homepage o al carrello con un massimo di due clic.

  Priorità: Alta.

- **Accesso rapido**: Gli utenti registrati devono poter accedere direttamente alle sezioni chiave (catalogo, carrello, ordini) 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: Alta.

- **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:** Durante i picchi stagionali, il sistema deve mantenere il funzionamento con almeno il 200% del carico normale, senza degradazione delle prestazioni superiore al 10%.

  Priorità: Bassa.

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.
  L'architettura a tre livelli garantisce la capacità, del sistema, di essere facilmente modificato, aggiornato o esteso.

  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.

- **Tracciabilità dei requisiti**: Il sistema deve garantire facile mappatura del codice nei requisiti.

  Priorità: Alta.

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 1.1

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 attuale

Il sistema attuale è basato su un modello MVC. La piattaforma attuale presenta diverse limitazioni che influiscono negativamente sull'efficienza operativa e sull'esperienza d'acquisto dei clienti, come la mancanza di un canale di comunicazione diretto e un processo di checkout lungo e frammentato.

3. Architettura software proposta

3.1. 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:

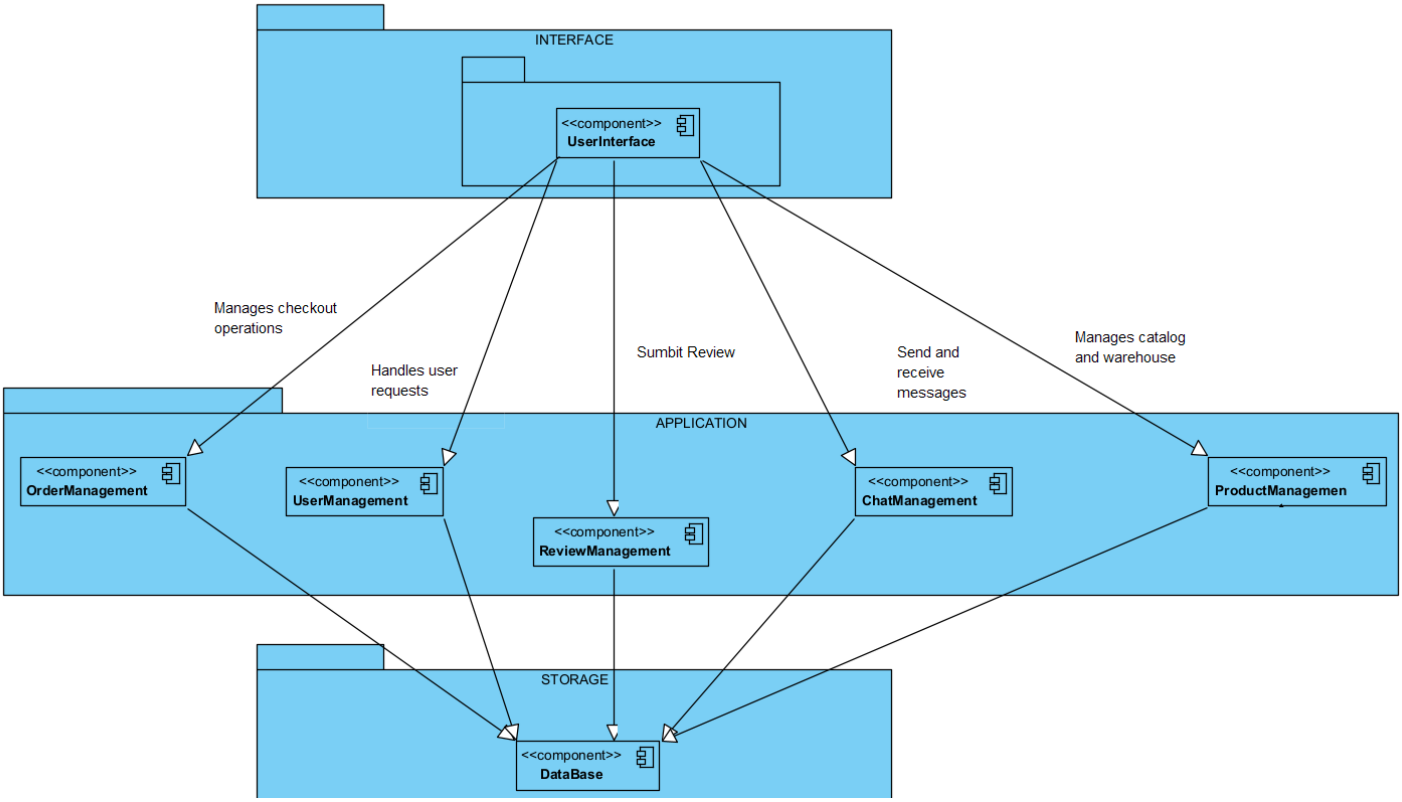
- **Presentation**, che si occupa di gestire l'interazione con l'utente;
- **Business**, che si occupa della logica applicativa;
- **Data Access Layer**, che si occupa della gestione della persistenza dei dati e della manipolazione degli stessi sul database.

Al processo di layering è stato abbinata la tecnica del partitioning, che ha individuato 4 moduli principali:

- la gestione degli utenti

- la gestione dell’acquisto
- la gestione del feedback
- la gestione del catalogo.

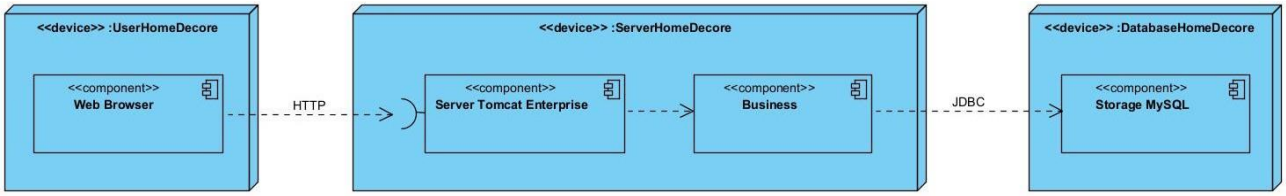
3.2. Decomposizione in sottosistemi



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 Chat**: Permette invio e ricezione di messaggi.

3.3. 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 Tomcat gestisce la logica applicativa e l'orchestrazione dei flussi.
- **Database Server**: MySQL conserva i dati persistenti (utenti, prodotti, ordini).

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

3.4. Gestione dei dati persistenti

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

❖ 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 backup e recupero sul DB.

3.5. Controllo degli accessi e sicurezza

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

3.6. Controllo globale del software

Al fine di offrire un approccio flessibile all'utente, per questo tipo di applicazione web si preferisce un controllo del software basato su eventi. Il web container si occuperà di distribuire le richieste provenienti dal client verso appositi controller, che a loro volta gestiranno la richiesta delegando le operazioni alle componenti di business.

3.7. Condizioni limite

Le boundary conditions relative alla gestione del sistema sono:

- **Installazione**: prima di avviare l'applicazione verrà definito come essa sarà accessibile, cioè tramite il server container Apache Tomcat Enterprise, che si basa sulla versione open source di Apache Tomcat, con componenti aggiuntive per garantire maggiore sicurezza e scalabilità dei servizi. L'installazione e configurazione del server Tomcat avverrà su una macchina remota, gestita da un'azienda specializzata nell'hosting e web service. Una volta attivato il server, verrà eseguito il deploy del sistema.

- **Avvio:** Dopo l'installazione, il sistema viene caricato sul server, durante l'avvio verranno inizializzate e caricate varie configurazioni come la connessione al DBMS MySQL, tramite l'utilizzo del driver derby JDBC e si verifica che tutto venga eseguito correttamente. Una volta completato l'avvio il sistema sarà accessibile dai vari utenti.

- **Manutenzione:** Eventuali aggiornamenti sul sistema verranno abilitati quando il sito sarà offline, verranno apportate le varie modifiche e successivamente il sistema sarà nuovamente avviato.

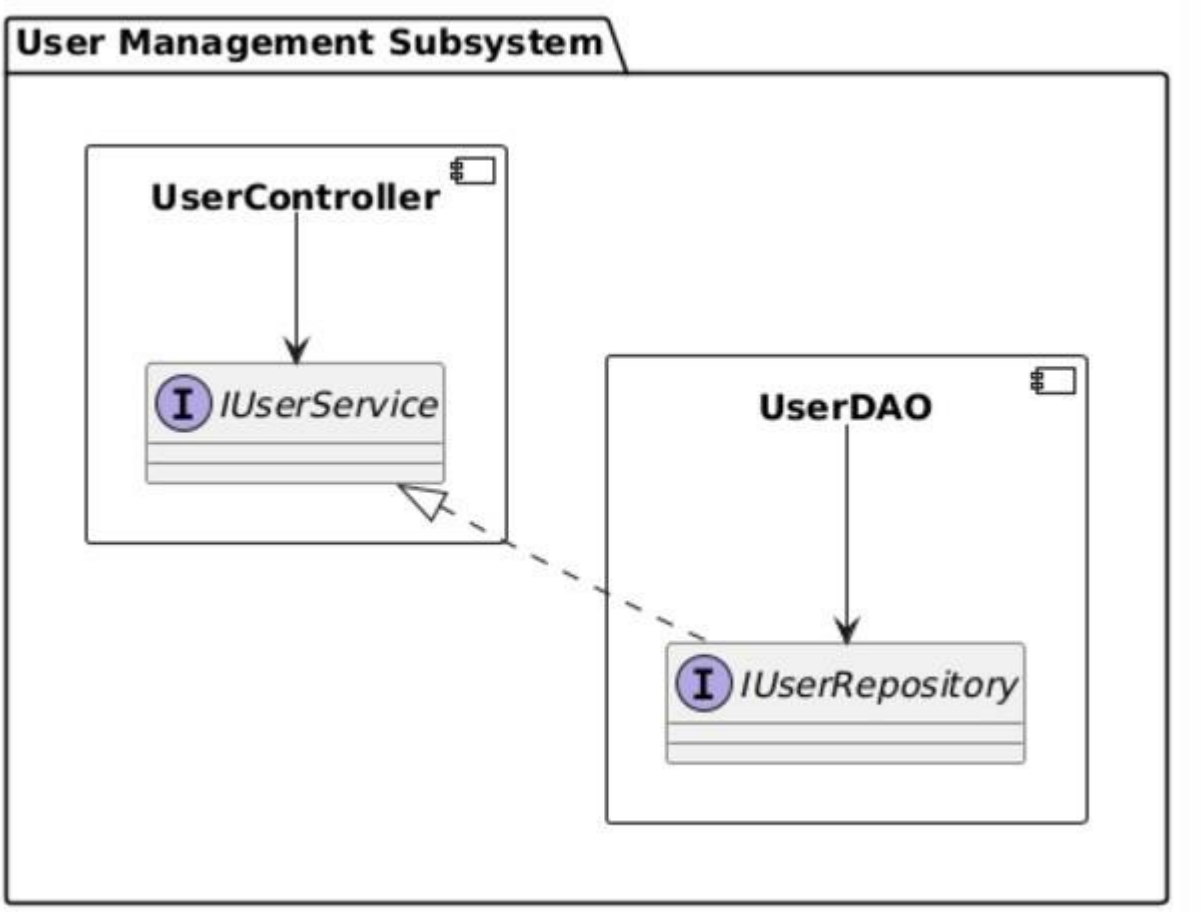
- **Fallimenti del sistema**: eventuali problemi relativi a crush del server o guasto della macchina remota saranno gestiti dall'azienda incaricata della gestione,nel caso alternativo cioè se il fallimento dipende da un'eccezione da noi non gestita(esempio,può capitare un problema realtivo all'aggiornamento del carrello che magari non aggiunge più di un prodotto), si provvederà alla sua risoluzione in modo permanente.

3.8. Servizi dei sottosistemi

Nota: i service vengono rappresentati come **interfacce** per favorire:

1. **Separazione delle responsabilità**: La logica applicativa non dipende da implementazioni specifiche ma solo da contratti definiti nelle interfacce.
2. **Facilità di testing**: È possibile fare mocking dei service durante i test, simulando comportamenti senza interazioni con DAO reali.
3. **Manutenibilità e flessibilità**: L'implementazione di un service può cambiare senza modificare i controller o i DAO che lo usano

3.8.1. Gestione Utenti

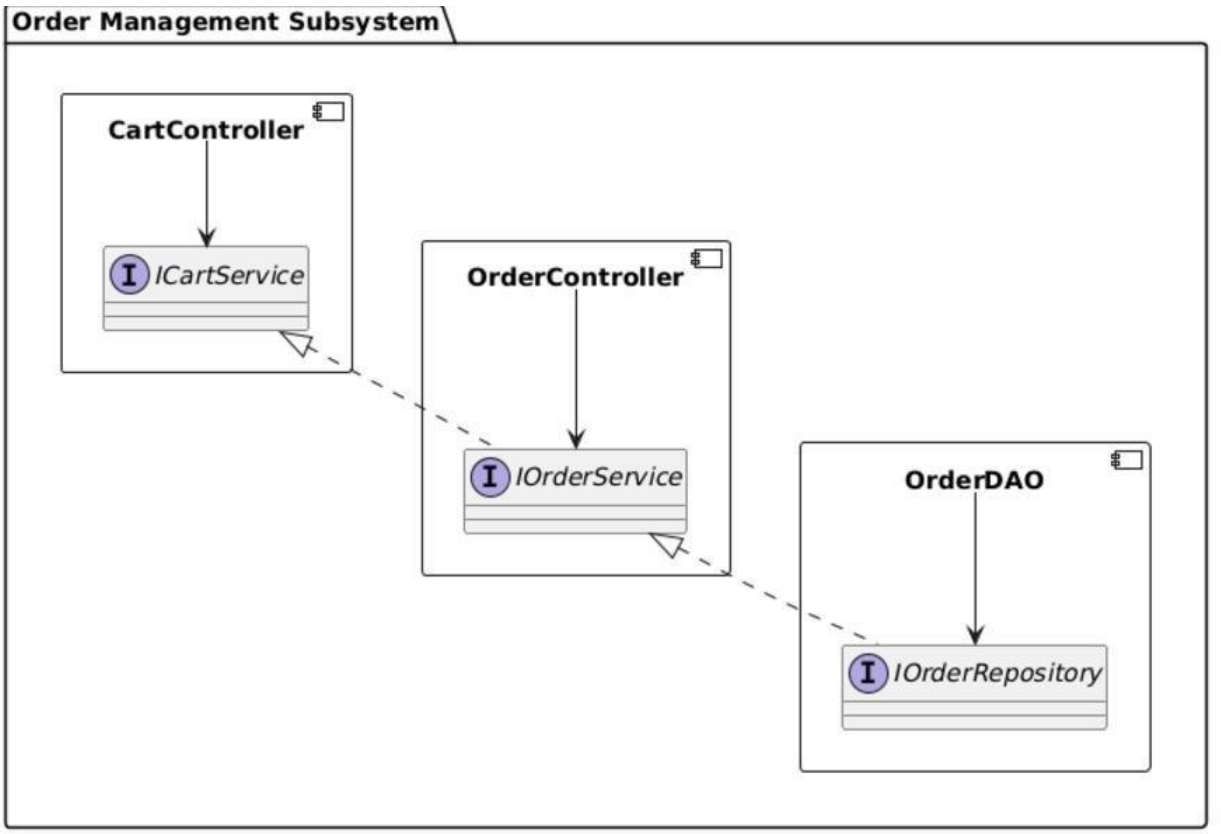


Il sottosistema User Management si occupa di registrazione, login e gestione del profilo.

- **UserController** rappresenta il punto di ingresso per le richieste degli utenti. Gestisce operazioni come la registrazione, il login.

- **IUserService** rappresenta l'interfaccia del livello di servizio (Service Layer). Qui si trovano le logiche applicative, come la validazione degli utenti.
- **UserDAO (Data Access Object)** è responsabile dell'accesso ai dati persistenti.
- **IUserRepository** definisce un'interfaccia per l'accesso ai dati del livello DAO.

3.8.2. Gestione Ordine



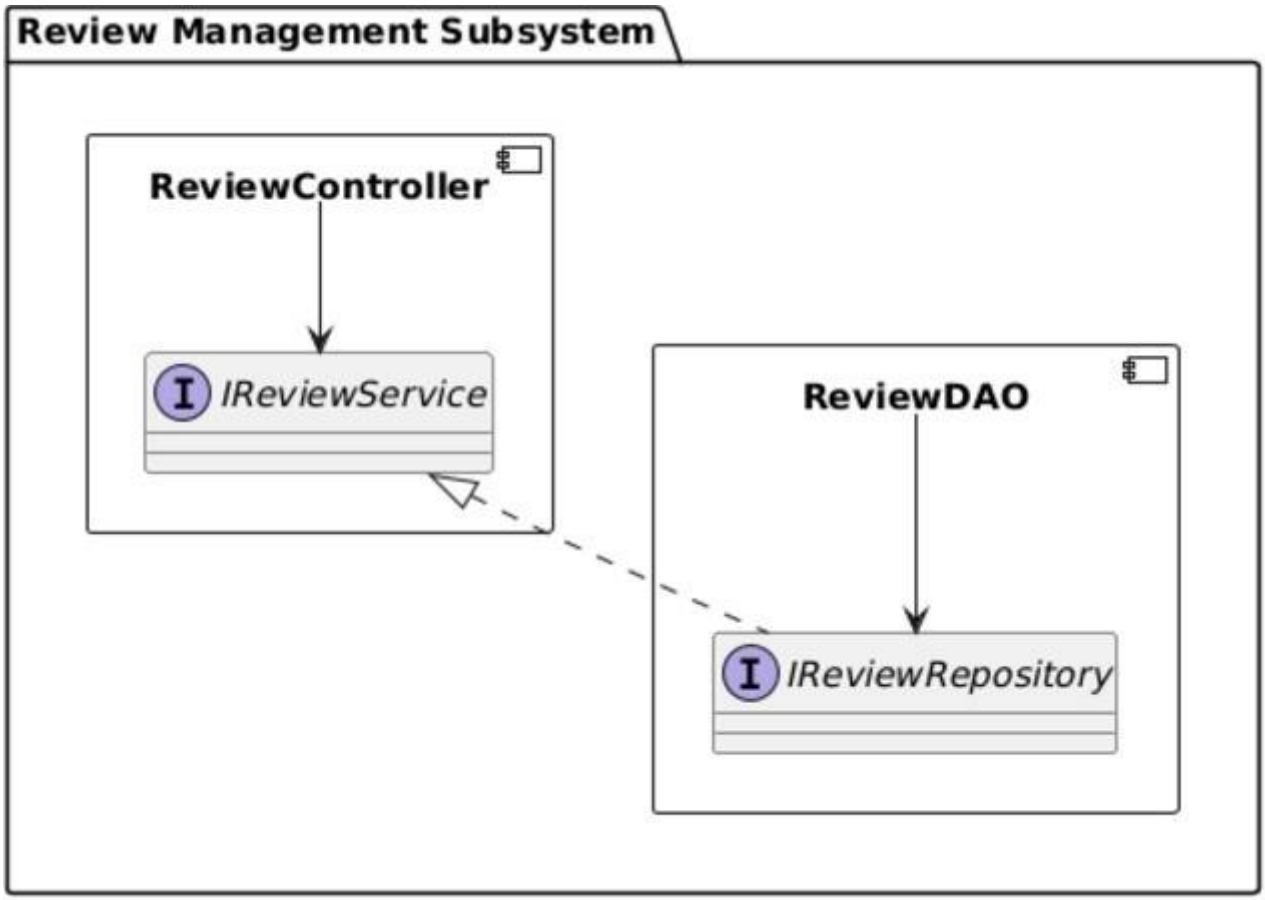
Questo sottosistema si occupa della gestione del carrello, del checkout e dello stato degli ordini.

- **CartController** e **OrderController** gestiscono l'interazione con gli utenti per il carrello e gli ordini.
- **ICartService** e **IOrderService** rappresentano i servizi logici che incapsulano il comportamento del carrello e degli ordini.
- **OrderDAO** fornisce l'accesso ai dati persistenti relativi agli ordini tramite

IOrderRepository

- **Carrello**: Consente agli utenti di aggiungere, modificare e rimuovere prodotti dal carrello.
- **Checkout**: Gestisce il processo di acquisto, inclusa la selezione del metodo di pagamento e l'inserimento delle informazioni di spedizione.
- **Stato degli ordini**: Permette agli utenti di visualizzare lo stato dei propri ordini.

3.8.3. Gestione Recensione

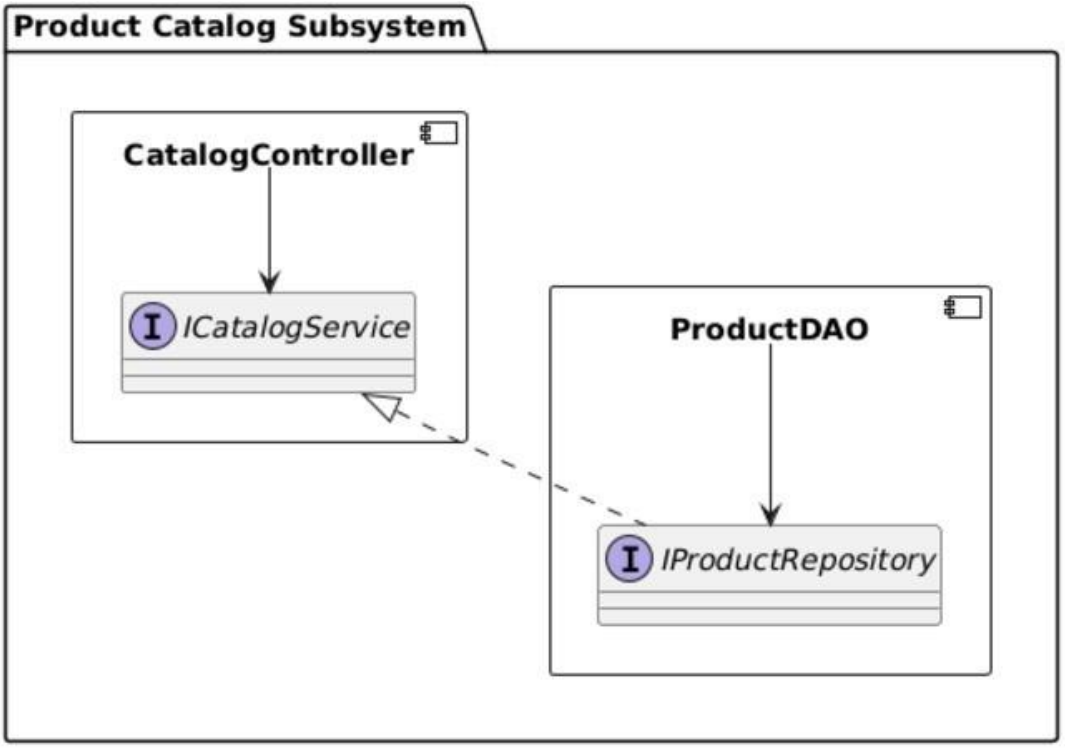


Questo sottosistema è responsabile di permettere agli utenti di lasciare recensioni e voti sui prodotti.

- **ReviewController** gestisce le richieste degli utenti per aggiungere o visualizzare recensioni.

- **IReviewService** è il livello intermedio che contiene la logica applicativa. Ad esempio, verifica che solo gli utenti autenticati possano recensire.
- **ReviewDAO** si occupa della comunicazione con il database per salvare o recuperare recensioni.
- **IReviewRepository** rappresenta l'interfaccia del livello DAO.

3.8.4. Gestione Prodotti

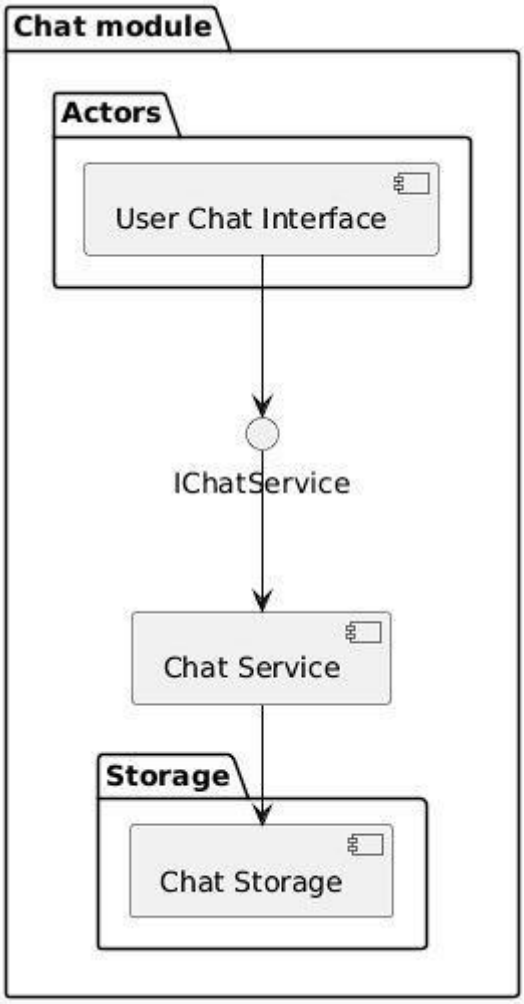


Questo sottosistema consente agli utenti di visualizzare e cercare i prodotti, mentre gli

amministratori possono gestirli.

- **CatalogController** elabora le richieste per la visualizzazione e modifica del catalogo.
- **ICatalogService** fornisce le funzionalità centrali del catalogo, come la ricerca avanzata o i filtri.
- **ProductDAO** è responsabile della persistenza dei dati relativi ai prodotti, accedendo al database tramite **IProductRepository**.

3.8.5. Gestione Chat



Questo sottosistema consente agli utenti di mandare messaggi al altri utenti tramite la chat.

- **User Chat Interface** permette all'utente di interfacciarsi con la chat. Permette di visualizzare messaggi e mettere a disposizione caselle per scrivere massaggi
- **Chat Service** elabora le richieste per la visualizzazione e invio del messaggio

- **IChatService** fornisce le funzionalità centrali del catalogo, come la ricerca avanzata o i filtri.

Chat Storage è responsabile della persistenza dei dati relativi ai messaggi,accedendo al database tramite **ChatService**.

-