Università degli Studi di Salerno
Corso di Ingegneria del Software

Pop!x
System Design Document
Versione 1.1

Data: 25/11/2024

Coordinatore del progetto:

Nome	Matricola
Scaparra Daniele Pio	0512116260

Partecipanti:

Nome	Matricola
Scaparra Daniele Pio	0512116260
Bonagura Grazia	0512116167
Nappi Antonio	0512117391
Nardiello Raffaele	0512118666

Scritto da:	Scaparra Daniele Pio, Bonagura Grazia, Nappi Antonio, Nardiello Raffaele

Revision History

Data	Versione	Descrizione	Autore
25/11/2024	1.0	Prima versione del SSD	Scaparra Daniele Pio, Bonagura Grazia, Nappi Antonio, Nardiello Raffaele
22/12/2024	1.1	Aggiunta descrizione dei sottosistemi	Scaparra Daniele Pio
22/12/2024	1.2	Riscritte le boundary conditions	Bonagura Grazia, Nappi Antonio
22/12/2024	2.0	Versione di rilascio con aggiunta servizi sottosistemi e glossario	Nardiello Raffaele, Scaparra Daniele Pio
06/01/2025	3.0	Ristrutturato SSD	Scaparra Daniele Pio, Bonagura Grazia, Nappi Antonio, Nardiello Raffaele

Indice

Outline

1. Introduzione

1.1 Scopo del sistema

Lo scopo del sistema è quello di sviluppare un e-commerce per la vendita esclusiva di Funko Pop!
Pop!x è una piattaforma web che si occupa di fornire un catalogo fornito e aggiornato di Funko Pop dei più disparati personaggi di brand che hanno ricevuto una propria versione funko.
Il target verso cui è destinato sarebbe della fascia d'età adolescenziale, ma questi prodotti stanno attirando sempre di più l'attenzione anche di un pubblico più adulto pronto a mettere mano al portafoglio per avere con sé le versione funko dei loro personaggi preferiti.
Gli utenti del sito sono gli utenti guest, ovvero chiunque può accedere al sito, ma che non acquista, in quanto non registrato, utenti registrati che possono effettuare la normale navigazione e comprare i prodotti a cui sono interessati e gli admin che gestiscono il catalogo, per mantenerlo sempre aggiornato, visualizzano gli ordini e gli utenti che si sono iscritti alla piattaforma web Pop!x.

1.2 Obiettivi di design

Durante la fase di System Design si è deciso di dare maggiore enfasi ai seguenti design goals.

1.2.1 Usabilità

Obiettivo: Garantire un'esperienza utente intuitiva e piacevole che faciliti la navigazione e l'uso del sito e-commerce, migliorando la soddisfazione degli utenti e massimizzando le operazioni effettuate dall'utente.

Requisiti chiave:

1. **Struttura della navigazione:**
    - Progettare un'interfaccia di navigazione chiara e ben organizzata che permetta agli utenti di trovare facilmente i prodotti desiderati, di accedere alle funzionalità messe a disposizione per quell'utente e di tornare facilmente alla homepage ovunque.

- Implementare un sistema di categorie e filtri che sia intuitivo e consenta una ricerca rapida ed efficiente.

2. **Design responsive:**
    - Assicurare che il sito sia responsive, fornendo un'esperienza ottimale su ogni device cui esso viene navigato, adattando quindi tutti i componenti affinché risultino ugualmente ben piazzati e gradevoli alla vista indipendentemente dallo schermo su cui sono mostrati.
    - Utilizzare framework e tecniche di responsive design per adattare il layout e il contenuto alle diverse dimensioni dello schermo.
3. **Design dell'interfaccia utente:**
    - Progettare un'interfaccia utente accattivante e coerente con l'identità del brand, utilizzando colori, font e stili visivi che rispecchiano il marchio e ciò che il sito intende promuovere, ovvero la vendita di Funko Pop.
    - Assicurare che tutti gli elementi interattivi siano facilmente identificabili e accessibili, con una chiara indicazione di cosa essi fanno.
4. **Funzionalità di ricerca:**
    - Implementare una funzionalità di ricerca efficiente e precisa, con filtri di ricerca utili, coerenti a ciò che lo store immagazzina per facilitare il reperimento dei prodotti e per raffinare il più possibile la ricerca.
    - Assicurarsi che i risultati della ricerca siano pertinenti e ordinati in modo logico.
5. **Feedback e gestione errori:**
    - Fornire feedback visivo immediato per le azioni degli utenti, come conferme di aggiunta al carrello, di completamento dell'acquisto, modifiche effettuate con successo, registrazioni e login andati a buon fine
    - Gestire gli errori in modo chiaro e comprensibile, offrendo soluzioni e guide per la risoluzione dei problemi, come ad esempio messaggi di errore chiari ed esplicativi, con suggerimenti per un corretto inserimento dei dati, ad esempio, o in generale per completare con successo la suddetta operazione.

1.2.2 Robustezza

Obiettivo: Garantire che la piattaforma e-commerce sia stabile, sicura e in grado di gestire situazioni impreviste, offrendo un'esperienza utente continua e affidabile.

Requisiti chiave:

1. **Stabilità tecnica:**
    - Implementare un'infrastruttura server affidabile, garantendo uptime elevato e riducendo al minimo i tempi di inattività.
    - Effettuare test rigorosi per identificare e correggere bug, utilizzando pratiche di sviluppo sicure e test unitari.
2. **Sicurezza:**
    - Proteggere i dati degli utenti utilizzando protocolli di sicurezza come HTTPS per le comunicazioni e tecniche di hashing per le password, come una versione ben aggiornata di SHA o BCrypt.
3. **Ottimizzazione delle prestazioni:**
    - Ottimizzare le query SQL per migliorare la performance del database e ridurre i tempi di risposta.
    - Monitorare e gestire l'uso delle risorse del server per prevenire colli di bottiglia e migliorare l'efficienza complessiva del sistema.
4. **Testing e manutenzione:**
    - Eseguire test continui e integrati per garantire che il sistema rimanga robusto anche con l'introduzione di nuove funzionalità.
    - Applicare aggiornamenti regolari al software per correggere vulnerabilità, migliorare le performance e aggiungere nuove funzionalità.

1.2.3 Manutenzione

Obiettivo: Assicurare che la piattaforma e-commerce rimanga aggiornata, efficiente e priva di bug attraverso pratiche di manutenzione regolari e proattive, migliorando continuamente l'esperienza utente e la sicurezza del sistema.

Requisiti chiave:

1. **Aggiornamenti regolari:**
    - Applicare aggiornamenti regolari al software di base (Java, Tomcat, MySQL) per correggere vulnerabilità, migliorare le performance e aggiungere nuove funzionalità.
    - Mantenere aggiornate le librerie e framework utilizzati, garantendo compatibilità e miglioramenti di sicurezza.
2. **Revisione codice:**
    - Eseguire revisioni del codice regolari per identificare e correggere potenziali problemi, migliorare la qualità del codice e garantire l'aderenza agli standard moderni.

3. **Documentazione:**
    - Tenere aggiornata la documentazione del sistema, includendo guide per sviluppatori, documentazione dell'architettura e manuali utente.
    - Mantenere registri delle modifiche dettagliati per documentare gli aggiornamenti, le correzioni di bug e le nuove funzionalità introdotte.

1.2.4 Affidabilità

Obiettivo: Garantire che la piattaforma e-commerce sia altamente affidabile, offrendo un'esperienza continua e senza interruzioni agli utenti, riducendo al minimo i tempi di inattività e garantendo l'integrità dei dati e delle transazioni.

Requisiti chiave:

1. **Tolleranza ai guasti:**
    - Progettare il sistema per degradare in modo graduale e gestibile piuttosto che fallire completamente in caso di problemi.
2. **Integrità dei dati:**
    - Assicurarsi che tutte le transazioni siano gestite correttamente, utilizzando tecniche di gestione delle transazioni come ACID (Atomicità, Coerenza, Isolamento, Durabilità) per garantire l'integrità dei dati.
    - Implementare robusti meccanismi di gestione degli errori per prevenire la perdita di dati e mantenere la coerenza del database.

1.3 Definizioni, abbreviazioni e acronimi

- *temporaneo:* questo si riferisce nello specifico nella sezione di controllo degli accessi al carrello usato dal guest che è temporaneo, nel senso che non persiste, in quanto l'utente non è registrato e sparisce, chiusa la sessione:
- *proprio:* si riferisce, sempre nel contesto della sezione del controllo degli accessi, quando si vuole indicare che quell'attore agisce sulla propria istanza della classe di riferimento con quell'operazione.
- *tutti:* si riferisce, sempre nel contesto della sezione del controllo degli accessi, quando si vuole indicare che quell'attore agisce su tutte le istanze della classe di riferimento con quell'operazione.
- *SDD*: si riferisce al System Design Document, un documento tecnico utilizzato per descrivere in dettaglio l'architettura, i componenti e le interazioni di un sistema software. Serve come una guida per gli sviluppatori, gli ingegneri e altri stakeholder, fornendo una visione chiara di come il sistema deve essere implementato.

- *RAD:* un documento che descrive in dettaglio i requisiti funzionali e non funzionali di un sistema. Questo documento è fondamentale nel ciclo di vita dello sviluppo del software, poiché serve come base per la progettazione, lo sviluppo e il testing del sistema.
- *Problem Statement:* è una dichiarazione chiara e concisa che definisce un problema specifico che un progetto, un processo o uno studio intende risolvere. È uno degli elementi fondamentali all'inizio di un progetto, poiché orienta l'attenzione su ciò che deve essere risolto e perché è importante affrontarlo.

1.4 Riferimenti

- Riferimento al Requirements Analysis Document
- Riferimento al Problem Statement.

1.5 Panoramica

In questo documento, il System Design Document (SDD), vengono mostrati i dettagli tecnici del design del sistema **Pop!x.**
Per una panoramica generale sul sistema si rimanda al documento del Problem Statement, mentre per quanto riguarda le funzionalità e i requisiti del sistema si rimanda, invece, al RAD (Requirement Analysis Document).
In questo documento troviamo un'introduzione all'architettura corrente del sistema, uno sguardo su quella proposta, vengono illustrati gli obiettivi di design che si intendono raggiungere, spiegati nel dettaglio attraverso i *requisiti chiave.*
Viene, inoltre, illustrata la suddivisione del sistema in sottosistemi, definendo il mapping Hardware/Software per assegnare ogni sottosistema ad uno specifico hardware.
Nel documento troviamo anche una descrizione del controllo degli accessi e i problemi di sicurezza, enfatizzando anche il controllo generale del software e le condizioni di contorno, ovvero le **Boundary conditions.**

2. Architettura software attuale

L'architettura software attuale è basata sulla suddivisione dei sottosistemi in tre strati principali, ognuno dei quali è responsabile della fornitura di specifiche funzionalità. Questo approccio garantisce una separazione delle responsabilità, migliorando la manutenibilità del sistema.
Abbiamo quindi:

1. ***Presentation Layer:*** è responsabile della visualizzazione delle pagine richieste dai client web. Questo strato include:

- **View Components**: componenti utilizzati per la visualizzazione dei dati, quindi JSP, HTML, CSS.
- **Controllers**: Java Servlets che ricevono le richieste dagli utenti, le elaborano e determinano quale vista restituire.
- **Client-Side Logic**: Eventuale logica client-side per migliorare l'interattività, come validazioni JavaScript e AJAX per aggiornamenti asincroni.

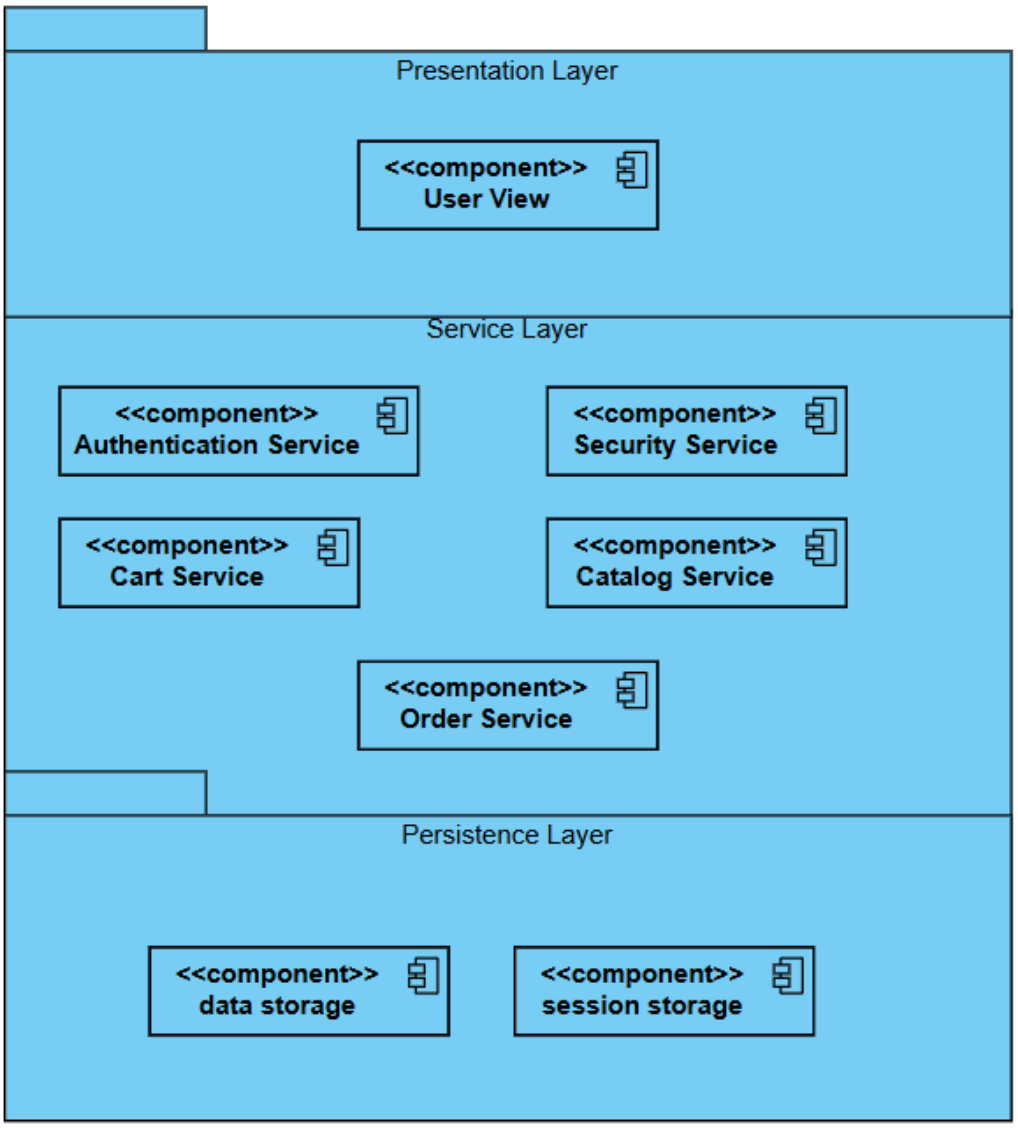
2. ***Service Layer*****:** media tra le richieste del client web e il Model (del modello architetturale MVC implementato) del sistema. Le sue responsabilità includono:
    - **Service Classes**: Classi che implementano la logica di business e le regole del sistema.
    - **Transaction Management**: Gestione delle transazioni per garantire che tutte le operazioni siano eseguite in modo coerente.
    - **Data Transfer Objects (DTOs)**: Oggetti utilizzati per trasferire dati tra il Presentation Layer e il Persistence Layer.
3. ***Persistence Layer*****:** si occupa della gestione dei dati persistenti del sistema. I suoi compiti sono:
    - **Data Access Objects (DAOs)**: Classi che forniscono metodi per interagire con il database MySQL.
    - **Database Connection Management**: Gestione delle connessioni al database, inclusi pool di connessioni per migliorare le performance.
    - **Sessioni:** si usano sessionId per mantenere informazioni volatili come il carrello o la persona loggata.

3. Architettura software proposta

3.1 Panoramica

A livello architetturale il sistema proposto e quello attuale coincidono.
L'obiettivo è quello di migliorare il sistema attuale, rimanendo invariate le scelte e l'implementazione architetturale già fatte.

3.2 Scomposizione in sottosistemi



Nel seguente component diagram possiamo distinguere i tre layer con i propri componenti.
Di seguito è riportata la descrizione di ogni layer, con i componenti che contiene e le dipendenze tra componenti

Presentation Layer:

- *User View*: è l'interfaccia degli utenti e rappresenta tutte le viste degli utenti (admin, user, gestori), quindi tutte le pagine che riguardano la visualizzazione

dei dati per pagine specifiche.

Service Layer:

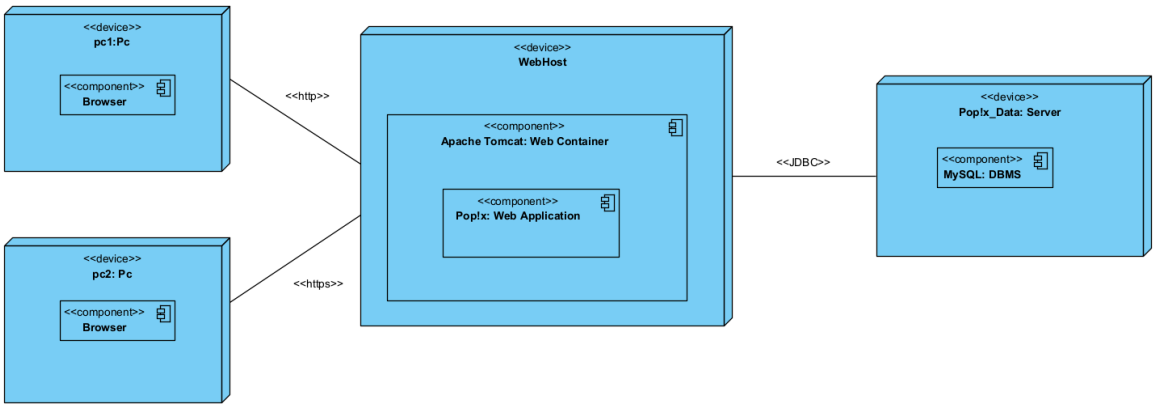
- *Authentication Service*: Responsabile dell'autenticazione e autorizzazione degli utenti. Verifica credenziali, controlla ruoli (admin, cliente, gestore ordine), e gestisce sessioni.
- *Security Service*: Garantisce la protezione dei dati e la gestione degli accessi con tecniche come l'hashing delle password.
- *Catalog Service*: Gestisce il catalogo e le operazioni da poter fare su di esso.
- *Cart Service*: Gestisce le operazioni sul carrello, come l'aggiunta, modifica e rimozione dei prodotti.
- *Order Service*: Gestisce gli ordini, tra cui creazione e aggiornamento dello stato.

Persistence Layer:

- *Data Storage*: Gestisce la memorizzazione e il recupero di informazioni sugli utenti, ordini, carrello, prodotti in maniera persistente.
- *Session Storage* : Gestisce la memorizzazione e il recupero di informazioni sugli utenti e carrello nella sessione.

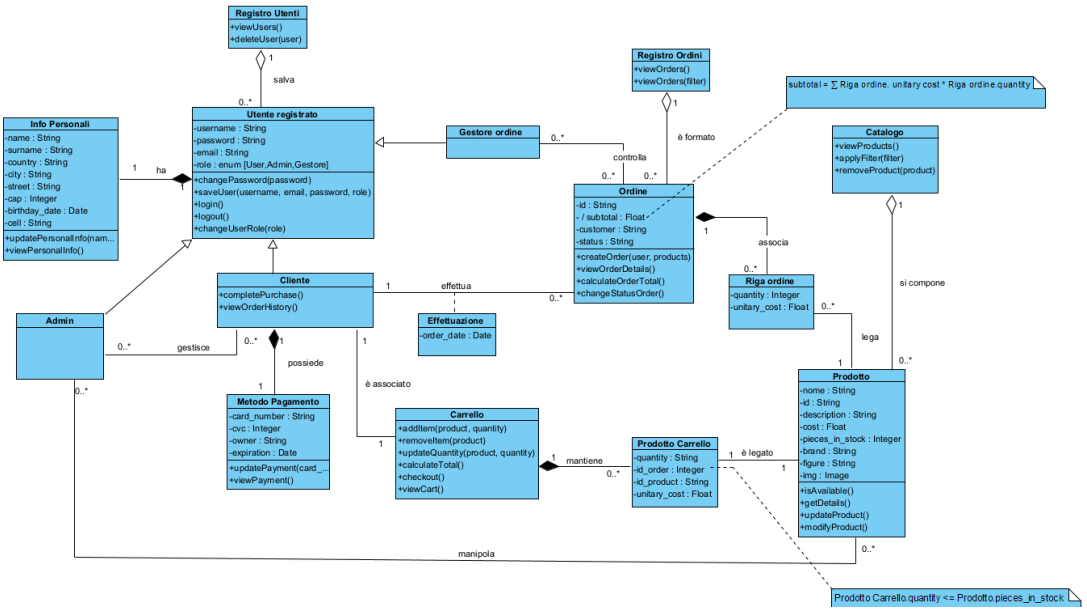
3.3 Mappatura Hardware/Software

Il sistema utilizza un'architettura client-server. Il Web Server è rappresentato da Apache Tomcat 9 che si trova su una singola macchina con sistema operativo Windows 11, viene utilizzato il pattern architetturale MVC, ove la logica del sistema è costituita dalle Servlets in Java, l'interfaccia utente (o view) è rappresentata da JSP (Java Servlet Page).
Il client è rappresentato dal Web browser usato dall'utilizzatore del sistema.
Abbiamo come protocolli di comunicazioni HTTP e HTTPS e la persistenza l'abbiamo con query in JDBC tra server e database.



3.4 Gestione Dati Persistenti

Class Diagram System Design



3.5 Controllo degli accessi e sicurezza

Oggetti / Attori	Catalogo	Ordini	Carrello
Cliente	visualizza	crea, visualizza (proprio)	aggiunge, rimuove, modifica, visualizza
Admin	aggiunge, modifica, elimina, visualizza (tutti)		
Gestore ordine	visualizza	visualizza gestisce stato (tutti)	
Guest	visualizza		aggiunge, rimuove, modifica (temporaneo)

Ogni utente del sistema avrà a disposizione una propria interfaccia grafica dedicata che gli permetterà di assolvere le funzionalità a lui assegnate.
L'interfaccia per utente e guest presenta caratteristiche comuni, in quanto l'utente guest è un utente non autenticato e che pertanto può accedere a quei servizi mediante un login.
L'admin e il gestore ordine, i quali rappresentano le figure amministrative del sistema, possono manipolare gli oggetti loro assegnati previa autenticazione.
A loro è inoltre vietato l'accesso al carrello e quindi alla funzione di acquisto, ma la visione del catalogo è disponibile per chiunque, come ovviamente vedere l'home page.
Ciò che cambia è la dashboard che offre ad ogni utente delle funzionalità ben specifiche per il proprio ruolo.
Anche la schermata di login è uguale per tutti, in quanto il sistema al riconoscimento delle credenziali inserite, riconoscerà che ruolo ha l'utente che si è appena autenticato e garantirà la corretta visualizzazione delle pagine che può visualizzare e l'accesso a quelle determinate funzionalità.

Le password e i dati di pagamento vengono criptati tramite hashing bcrypt.

3.6 Controllo Software Globale

Per quanto riguarda il controllo del software globale abbiamo la componente **Server** che gestisce le varie richieste del **Client**. Tramite le classi **Java Servlet** il Server smista le richieste del client e successivamente ne delega esecuzione ed elaborazione della risposta.
Talvolta si rende necessaria l'interazione con il **DBMS** nel recuperare, salvare o modificare elementi nel **Database,** questo viene reso possibile dalle classi **Dao** e **Beans**. La visualizzazione delle pagine viene gestita dalle **JSP** *(Java server pages)*.
Per quanto riguarda le richieste il **Web Container** *(Apache Tomcat)* vengono integrati i protocolli HTTP e HTTPS.

3.7 Boundary conditions

Errore di autenticazione durante il checkout

- **Descrizione:** Se un utente non autenticato tenta di completare un acquisto, il sistema deve richiedere il login o la registrazione.
- **Condizioni di ingresso:**
    - L'utente ha articoli nel carrello e accede alla schermata di checkout senza essere autenticato.
- **Condizioni di uscita:**
    - L'utente è reindirizzato alla pagina di login/registrazione, con un messaggio che spiega il problema.
    - Gli articoli rimangono salvati nel carrello, pronti per essere acquistati dopo l'accesso.

Errore durante il pagamento

Descrizione: Se si verifica un errore durante il pagamento (ad esempio, carta rifiutata), il sistema deve garantire che l'ordine non venga registrato come completato e notificare l'utente.

- **Condizioni di ingresso:**

    - L'utente ha avviato il processo di pagamento fornendo i dettagli richiesti.
    - Il gateway di pagamento restituisce un errore (es. carta rifiutata, connessione persa).
- **Condizioni di uscita:**
    - L'utente è informato dell'errore tramite un messaggio chiaro e suggerimenti per risolverlo (es. verificare i dettagli della carta o provare con un altro metodo di pagamento).
    - Il carrello dell'utente rimane intatto, consentendo un nuovo tentativo di pagamento.

Errore di aggiornamento dei dettagli utente

Descrizione: Se un utente tenta di aggiornare i propri dettagli (ad esempio, email o password) e si verifica un errore, il sistema deve assicurare che le modifiche non salvate non compromettano i dati esistenti.

- **Condizioni di ingresso:**
    - L'utente accede alla sezione di gestione del profilo e modifica i dati personali.
    - Un errore (es. connessione persa o problema con il database) impedisce il salvataggio.
- **Condizioni di uscita:**
    - L'utente è notificato con un messaggio chiaro che specifica l'errore.
    - I dati esistenti non vengono modificati.
    - Le modifiche proposte vengono salvate temporaneamente, consentendo un nuovo tentativo.

Errore di disponibilità del prodotto

Descrizione: Se un utente tenta di acquistare un prodotto che non è più disponibile (es. esaurito durante il checkout), il sistema deve gestire la situazione in modo trasparente.

- **Condizioni di ingresso:**
    - L'utente aggiunge un prodotto al carrello e procede al checkout.
    - Il prodotto viene venduto ad un altro utente prima del completamento dell'acquisto.
- **Condizioni di uscita:**
    - Il sistema informa l'utente che il prodotto non è più disponibile.
    - L'utente è invitato a rimuovere il prodotto o a iscriversi a una lista di attesa.

Errore durante la modifica degli ordini

Descrizione: Se un utente o un amministratore modifica un ordine in lavorazione (es. cambio di indirizzo o aggiunta di articoli) e si verifica un errore, il sistema deve assicurarsi che lo stato dell'ordine rimanga coerente.

- **Condizioni di ingresso:**
  - Un ordine è in fase di elaborazione, e un utente o un admin tenta di modificarne i dettagli.
- **Condizioni di uscita:**
  - L'utente è notificato che la modifica non è stata completata.
  - Lo stato dell'ordine originale è mantenuto senza alterazioni.

4. Servizi del Sottosistema

Elenco in forma tabellare dei servizi del Service Layer con nome del service component (la parola Service vicino ogni component è omessa per evitare ripetizioni) e servizi offerti.

Service Component	Servizi Offerti
Authentication	- Verifica delle credenziali utente - Gestione login/logout - Assegnazione ruoli (utente, admin, ecc.)
Catalog	- Gestione dei dettagli dei prodotti (nome, descrizione, prezzo, stock) - Organizzazione in categorie e filtri
Security	- Protezione dei dati sensibili (es. hashing delle password) - Controllo degli accessi ai sottosistemi
Validation	- Controllo della validità degli input utente (es. dati del profilo, dettagli di pagamento)

	- Validazione dei dati durante il checkout
Cart	- Aggiunta, rimozione e modifica degli articoli nel carrello - Visualizzazione del contenuto del carrello
Order	- Creazione di nuovi ordini - Aggiornamento dello stato degli ordini - Tracciamento degli ordini e cronologia

Glossario

Termine	Descrizione
ACID	Proprietà delle transazioni nei database che garantiscono: Atomicità (tutte le operazioni sono eseguite o annullate), Coerenza (lo stato del database rimane valido), Isolamento (transazioni indipendenti) e Durabilità (persistenza dei dati).
AJAX	Asynchronous JavaScript and XML: tecnica per aggiornare parti di una pagina web senza ricaricare l'intera pagina, migliorando la reattività e l'interattività delle applicazioni web.
Beans	Componenti Java che incapsulano dati e logiche, seguendo convenzioni come l'uso di metodi getter e setter per accedere agli attributi.
Boundary Conditions	Condizioni limite o scenari eccezionali che il sistema deve gestire correttamente per garantire affidabilità e robustezza (es. errori di pagamento, prodotti non disponibili).
Client-Side Logic	Logica implementata lato client (es. browser) per migliorare l'interattività e

	ridurre il carico sul server, spesso utilizzando linguaggi come JavaScript e tecnologie come AJAX.
DAO	Data Access Object: pattern di progettazione che separa l'accesso ai dati dalla logica di business, semplificando la gestione e rendendo il codice più manutenibile.
Database	Raccolta organizzata di dati che possono essere facilmente gestiti, aggiornati e interrogati.
DBMS	Database Management System: software per creare, gestire e interrogare database in modo efficiente e sicuro.
Design Goals	Obiettivi di progettazione che guidano lo sviluppo del sistema, includendo aspetti come usabilità, affidabilità, robustezza e manutenibilità.
Framework	Insieme di strumenti e librerie che forniscono un'architettura base per sviluppare software, spesso seguendo best practices e design patterns
Java Servlet	Classe Java che gestisce richieste e risposte HTTP lato server, utilizzata per creare applicazioni web dinamiche
JavaScript	Linguaggio di programmazione per il web, utilizzato per creare interazioni dinamiche lato client e migliorare l’esperienza utente.
JSP	JavaServer Pages: tecnologia per generare contenuti web dinamici combinando codice Java ed elementi HTML.
MVC	Model-View-Controller: pattern architetturale che separa la logica di business (Model), la gestione degli input (Controller) e la presentazione (View).

Web Container	Componente software che gestisce servlet e JSP, facilitando l'interazione tra client e server attraverso protocolli come HTTP e HTTPS.