Università degli Studi di Salerno
Corso di Ingegneria del Software

Pop!x
System Design Document
Versione 1.0

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

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.
6. **Consistenza dell'esperienza dell'utente: #usabilità**
    - Assicurare che tutte le funzionalità del sito siano sempre disponibili e funzionanti, riducendo al minimo i tempi di inattività e i periodi di manutenzione.
    - Fornire informazioni chiare agli utenti in caso di problemi, incluse pagine di errore informative.

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. **Backup dei dati:**
   - Eseguire backup regolari dei dati, includendo sia il database MySQL che i file del codice sorgente, per garantire il recupero rapido in caso di perdita di dati.
   - Verificare periodicamente i backup per assicurarsi che siano completi e ripristinabili.
4. **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 i **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:

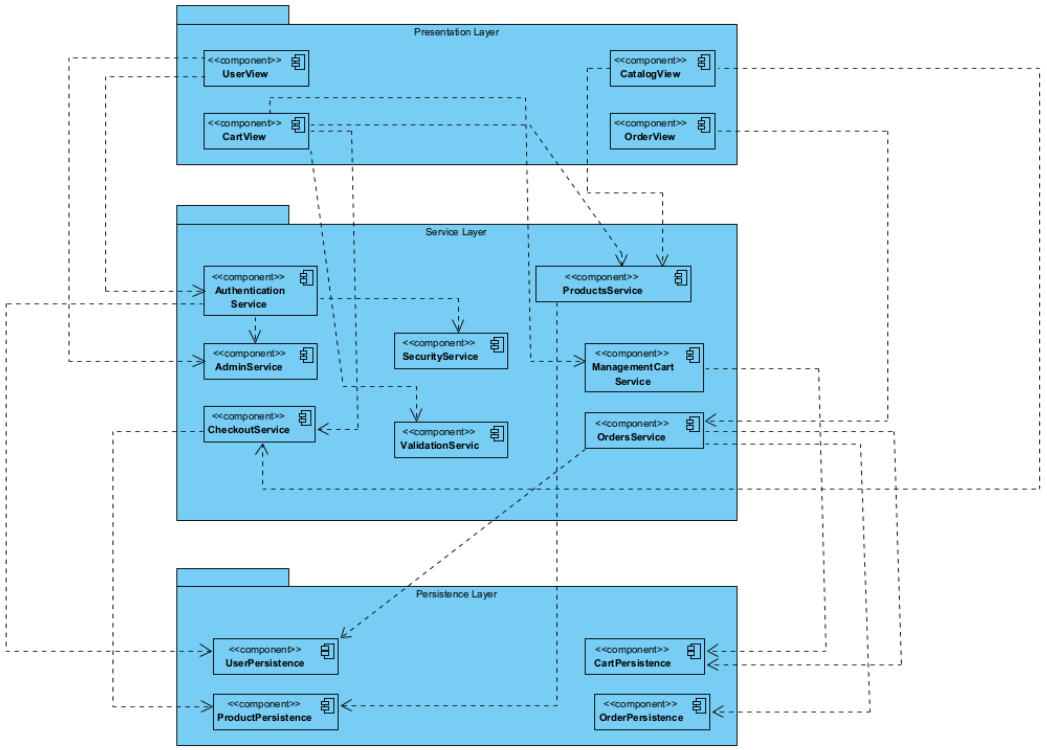
1. ***Presentation Layer:*** è responsabile della visualizzazione delle pagine richieste dai client web. Questo strato include:
    - **View Components**: componenti utilizzati per la visualizzazione dei dati, quindi JSP, HTML, CSS.
    - **Controllers**: Java Servlets che ricevono le richieste dagli 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 pattern usato MVC) 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.

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.

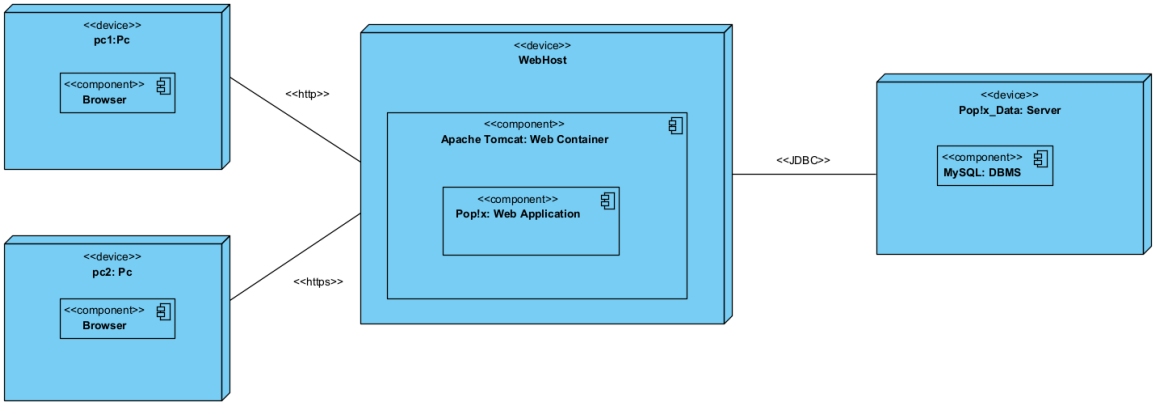
3.2 Scomposizione in sottosistemi



3.3 Mappatura Hardware/Software

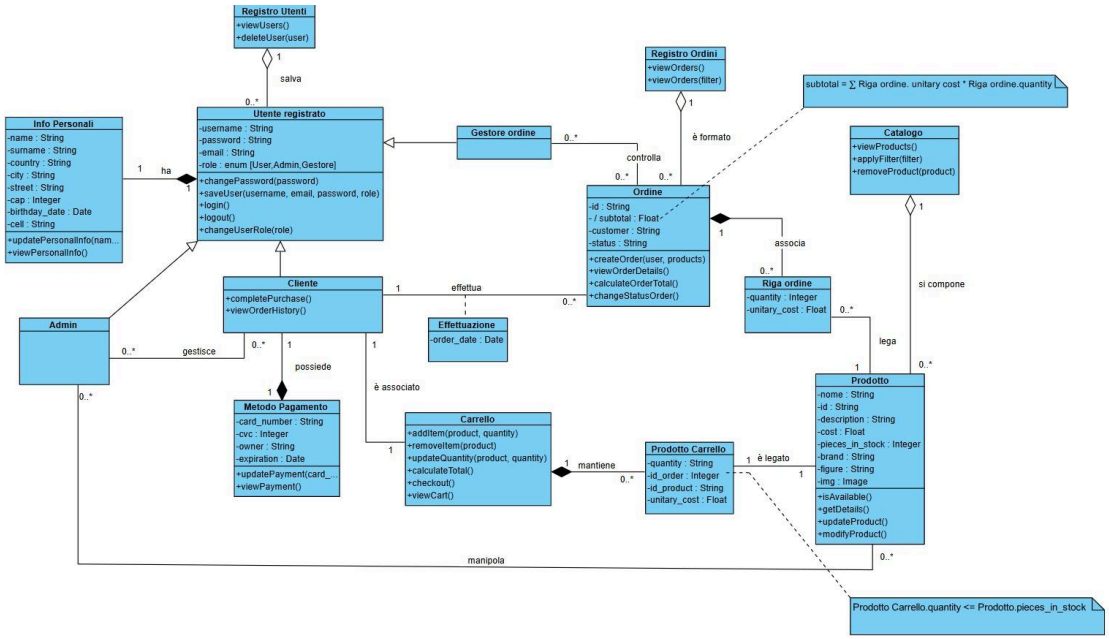
Il sistema utilizza un'architettura client-server. Il Web Server è rappresentato da Apache Tomcat 9 che si trova su una singola macchina con sistema operativo Windows 11, viene utilizzato il pattern architetturale MVC, ove la logica del sistema è costituita dalle Servlets in Java, l'interfaccia utente (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	Utenti	Ordini	Carrello
Cliente	visualizza	modifica profilo (proprio)	crea, visualizza (proprio)	aggiunge, rimuove, modifica, visualizza
Admin	aggiunge, modifica, elimina, visualizza (tutti)	visualizza, promuove, elimina (tutti)	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

Boundary Condition: Gestione degli errori dopo l'acquisto

Descrizione:
Questa condizione definisce i limiti del sistema in relazione alla gestione degli errori che possono verificarsi durante o subito dopo il processo di acquisto di un prodotto.

1. **Errore di caricamento della pagina dopo l'acquisto**:
    - Il sistema deve garantire che la pagina di conferma dell'acquisto venga caricata correttamente dopo il completamento dell'ordine. In caso di errore di caricamento (ad esempio, timeout del server o errore di rete), il sistema deve restituire un messaggio di errore chiaro all'utente, indicando che l'acquisto è stato effettuato correttamente, ma la pagina non è riuscita a caricarsi.
2. **Errore del carrello non aggiornato dopo l'acquisto**:
    - Dopo il completamento dell'acquisto, il sistema deve aggiornare lo stato del carrello dell'utente per riflettere che gli articoli sono stati acquistati. Se il carrello non viene aggiornato correttamente, il sistema deve verificare

periodicamente lo stato dell'ordine e aggiornare il carrello in modo asincrono, notificando all'utente se l'operazione di aggiornamento non è riuscita.

Condizioni di ingresso:

- L'utente ha completato il processo di acquisto e ha fornito le informazioni necessarie (dati di pagamento, indirizzo di spedizione).
- Il sistema è connesso ai servizi necessari per l'aggiornamento del carrello e la gestione delle sessioni utente.

Condizioni di uscita:

- L'utente è correttamente informato degli eventuali errori e ha accesso a istruzioni chiare per risolvere il problema.
- L'ordine è stato registrato correttamente, ma eventuali errori di interfaccia utente o carico della pagina non influiscono sul completamento dell'acquisto.

Azioni Preventive:

- **Timeout delle operazioni**: Implementare dei timeout ragionevoli per le operazioni critiche, come il caricamento della pagina di conferma e l'aggiornamento del carrello, con una gestione efficace degli errori.
- **Ridondanza dei dati**: Utilizzare un sistema di salvataggio ridondante per i dettagli dell'ordine per evitare la perdita di informazioni in caso di errori di rete o problemi server-side.
- **Log degli eventi**: Registrare in modo dettagliato ogni fase del processo di acquisto, incluso il caricamento della pagina di conferma e l'aggiornamento del carrello, in modo da avere un tracciamento preciso degli eventuali errori riscontrati.
- **Monitoraggio e notifiche**: Integrare un sistema di monitoraggio che possa notificare ai tecnici eventuali errori critici, come problemi di aggiornamento del carrello o di autenticazione, per un intervento tempestivo.

Metriche di Performance:

- **Tempo di caricamento della pagina di conferma**: Monitorare il tempo medio necessario per caricare la pagina di conferma dopo l'acquisto, identificando eventuali rallentamenti o anomalie.
- **Tasso di fallimento nell'aggiornamento del carrello**: Calcolare il numero di errori di aggiornamento del carrello rispetto al totale degli acquisti effettuati.
- **Occorrenza di errori "utente non trovato"**: Analizzare la frequenza con cui si verificano problemi di autenticazione durante il processo di acquisto, verificando se ci sono pattern comuni che potrebbero essere migliorati.

Risultati attesi:

- Gli utenti devono essere in grado di completare un acquisto anche se si verifica un problema di caricamento della pagina di conferma, grazie a notifiche chiare e al salvataggio dell'ordine nel database.
- Il carrello deve essere aggiornato correttamente nel 99% dei casi entro un tempo prestabilito, senza necessità di intervento manuale.

- Gli utenti devono ricevere istruzioni dettagliate e immediate in caso di problemi di autenticazione, riducendo al minimo l'impatto sull'esperienza di acquisto.

4. Servizi del Sottosistema

Glossario

Server	Sistema hardware o software che fornisce servizi, risorse o dati ad altri computer, chiamati client, attraverso una rete (come Internet o una rete locale). Il termine "server" può riferirsi sia a un dispositivo fisico (hardware) sia a un software specifico che esegue operazioni di "servizio".
Client	Un client è un dispositivo o un software che invia richieste a un server per accedere a dati, servizi o risorse. Il client utilizza una rete per comunicare con il server, che risponde fornendo le informazioni richieste. In altre parole, il client è l'entità che avvia una comunicazione, mentre il server è responsabile di rispondere a quella richiesta.
Java Servlet	Una Java Servlet è una classe Java che viene eseguita su un server web o un server di applicazioni per gestire richieste da parte dei client, di solito tramite il protocollo HTTP. Le servlet sono utilizzate principalmente per creare applicazioni web dinamiche, generando contenuti web in base a richieste specifiche inviate dai client (come i browser).
DBMS	Un DBMS (Database Management System) è un software che consente la creazione, gestione, manipolazione e organizzazione di database, permettendo agli utenti di archiviare, modificare, estrarre e gestire dati in modo efficiente. Il DBMS fornisce strumenti per la gestione dei dati, garantisce la loro integrità, sicurezza e consente di effettuare query per ottenere informazioni utili.
Database	Un database è una raccolta organizzata di dati che vengono memorizzati e gestiti in modo da poter essere facilmente accessibili, aggiornati e manipolati. I dati in un database possono essere strutturati in modo

	sistematico, utilizzando tabelle, colonne e righe nel caso di database relazionali, o in documenti, grafi e altri formati nei database non relazionali. Lo scopo principale di un database è quello di fornire un sistema efficiente per la gestione delle informazioni, consentendo di effettuare ricerche, recuperare dati specifici e archiviare grandi volumi di informazioni in modo sicuro e organizzato.
Jsp	JSP (JavaServer Pages) è una tecnologia utilizzata per sviluppare applicazioni web dinamiche basate su Java. Consente di creare pagine web in cui il contenuto dinamico, generato dal server, può essere inserito all'interno di una pagina HTML. JSP permette di mescolare codice Java con markup HTML utilizzando speciali tag e direttive. Quando una richiesta arriva al server, il codice Java contenuto in una pagina JSP viene eseguito per generare il contenuto dinamico, che viene poi inviato al client sotto forma di una normale pagina HTML. Questa tecnologia semplifica la creazione di contenuti dinamici, separando la logica di business dal design della pagina.
Dao	Il Data Access Object, abbreviato come DAO, è un pattern di progettazione utilizzato in ambito software per separare la logica di accesso ai dati dalla logica di business dell'applicazione. Il DAO agisce come un'interfaccia tra l'applicazione e la base di dati, consentendo di eseguire operazioni di lettura, scrittura, aggiornamento e cancellazione dei dati senza dover conoscere i dettagli specifici del database sottostante. Questo pattern facilita la manutenzione del codice, rendendo più semplice sostituire o modificare il database senza impattare le altre parti del sistema, e supporta una gestione più ordinata e modulare del codice.
Beans	Un bean è un componente software che incapsula dati e comportamenti in un'applicazione Java, tipicamente utilizzato in contesti di programmazione orientata agli oggetti. In particolare, i JavaBeans sono classi che seguono convenzioni specifiche, come l'uso di costruttori senza argomenti, metodi getter e setter per accedere ai dati e la possibilità di essere istanziati, configurati e gestiti tramite un contenitore, come un container di servlets o un framework di gestione delle dipendenze. I JavaBeans sono spesso usati per rappresentare entità o oggetti di valore all'interno di un'applicazione, come nel caso di trasferimento di dati tra livelli (ad esempio, tra un layer di business e uno di

	presentazione).
Web Container	Un web container è un componente software che gestisce l'esecuzione di applicazioni web basate su Java. Il suo compito principale è quello di ricevere le richieste HTTP dal client, inoltrarle alle servlet o alle JSP (JavaServer Pages) appropriate, e restituire una risposta al client. Il web container fornisce anche il supporto per la gestione delle sessioni, l'esecuzione di servlet, l'inizializzazione e la distruzione delle risorse, e la gestione delle configurazioni per le applicazioni web. Inoltre, si occupa della sicurezza e della gestione della concorrenza per garantire che più utenti possano interagire con l'applicazione senza conflitti. Esempi di web container includono Apache Tomcat, Jetty e GlassFish.
JavaScript	JavaScript è un linguaggio di programmazione utilizzato principalmente per creare contenuti web dinamici. Viene eseguito lato client, cioè nel browser dell'utente, e consente di modificare l'aspetto e il comportamento delle pagine web in tempo reale. JavaScript può essere utilizzato per rispondere a eventi come clic, invii di form o movimenti del mouse, aggiornando il contenuto della pagina senza bisogno di ricaricarla. Oltre a manipolare elementi HTML e CSS, JavaScript può interagire con server web per inviare e ricevere dati tramite tecnologie come AJAX. È un linguaggio fondamentale per lo sviluppo web, spesso usato insieme a HTML e CSS per costruire applicazioni web interattive.
Ajax	AJAX (Asynchronous JavaScript and XML) è una tecnica di sviluppo web che consente di aggiornare una pagina web in modo asincrono, cioè senza dover ricaricare l'intera pagina. Utilizzando JavaScript, AJAX permette di inviare e ricevere dati dal server in background e aggiornare solo una parte della pagina. Questo approccio migliora l'esperienza utente, rendendo le applicazioni web più veloci e interattive, poiché evita di dover caricare nuovamente l'intera pagina per ogni azione dell'utente. Sebbene il nome AJAX faccia riferimento a XML, oggi viene comunemente utilizzato con formati come JSON per scambiare dati tra il client e il server.
Framework	Un framework è un insieme strutturato di strumenti, librerie e linee guida che semplificano lo sviluppo di software, fornendo una base predefinita su cui costruire

	applicazioni. In pratica, un framework offre un'architettura di partenza che consente agli sviluppatori di concentrarsi su aspetti specifici della propria applicazione, come la logica di business, senza dover riscrivere codice di base per operazioni comuni come la gestione delle richieste, l'accesso ai dati o la sicurezza. I framework sono spesso progettati per un determinato tipo di applicazione, come le applicazioni web, le app mobili o le interfacce utente, e possono includere funzioni di base, convenzioni di codice e best practices. Esempi comuni di framework sono Django per Python, React per JavaScript e Spring per Java.