Università degli Studi di Salerno
Corso di Ingegneria del Software

BirdComics
System Design Document
Versione 1.0

BirdComics

Data: 24/11/2024

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
De Rosa Giuseppe	0512110683
Buccino Adriano	0512120484

Scritto da:	

Revision History

Data	Versione	Descrizione	Autore

1. INTRODUZIONE **4**
1.1. scopo del sistema 4
1.2. Obiettivi di progettazione 4
1.2.1. Prestazioni 4
1.2.2. Garanzia di funzionamento 5
1.2.3. Manutenzione 5
1.2.4. Utente finale 6
1.2.5. Compromessi 6
1.3. Definizioni, acronimi e abbreviazioni 6
1.4. Riferimenti 7
1.5. Panoramica 7
2. Architettura attuale **7**
3. Architettura software proposta **7**
3.1. panoramica 7
3.2. decomposizione del sottoproblema 7
3.3. Mappatura hardware/software 9
3.4. Gestione dei dati persistenti 10
3.5. Controllo degli accessi e sicurezza 10
3.6. Controllo software globale 12
3.7. Condizioni boundary 12
Primo avvio 12
Avvio in caso di fallimento 13
Errato accesso ai dati persistenti 13

1. INTRODUZIONE

1.1. scopo del sistema

Il sistema ha lo scopo di sviluppare e gestire una piattaforma di e-commerce dedicata alla vendita di fumetti, denominata *BirdComics*. La creazione di un e-commerce focalizzato sulla vendita di fumetti, si rivela una risposta appropriata alle necessità di un mercato in continua crescita. Differenziandosi nettamente da un negozio fisico. Un aspetto fondamentale di questa iniziativa è l'accessibilità: un e-commerce è aperto 24 ore su 24, 7 giorni su 7, permettendo ai consumatori di effettuare acquisti in qualsiasi momento, senza le restrizioni di orario imposte dai negozi tradizionali. Questa piattaforma permette agli utenti di acquistare fumetti in formato cartaceo, fornendo maggiore flessibilità e comodità esplorando una vasta selezione di titoli (dai grandi classici ai fumetti in edizione limitata o rari da trovare) che un negozio online può offrire rispetto a uno spazio fisico, il quale è limitato dalla capacità espositiva. L'obiettivo principale è offrire un'esperienza di acquisto comoda, accessibile e personalizzata, raggiungendo un pubblico globale e soddisfacendo le crescenti esigenze dei lettori di fumetti. Inoltre, il sistema dovrà essere altamente scalabile, permettendo di gestire un ampio catalogo di prodotti, utenti e transazioni senza compromettere le performance.

1.2. Obiettivi di progettazione

1.2.1. Prestazioni

Tempo di risposta	Il sistema deve rispondere a ogni richiesta dell'utente (come la navigazione o l'acquisto) entro 2 secondi (**RNF7**).
Throughput	Il sistema deve supportare almeno 1000 utenti simultanei senza degradare le prestazioni (**RNF8**).
Memoria	L'uso della memoria deve essere ottimizzato per mantenere prestazioni elevate garantendo il tempo di risposta ed il throughput sopra citati
Velocità ricerca	I risultati della ricerca nel catalogo prodotti devono essere aggiornati in tempo reale senza ricaricare la pagina (**RNF9**).

1.2.2. Garanzia di funzionamento

Robustezza	Il sistema deve resistere a input non validi e a comportamenti imprevisti degli utenti senza andare in crash.
Affidabilità	Il sistema deve garantire un funzionamento stabile, con una disponibilità del 99.9% (**RFN5**).
Disponibilità	Il sito deve essere operativo quasi costantemente, con downtime ridotto al minimo per manutenzione
Tolleranza ai guasti	Deve continuare a funzionare anche in caso di errori, ad esempio grazie a server di backup
Sicurezza	Protezione contro accessi non autorizzati (**RNF1**), password crittografate (**RNF2**), prevenzione di attacchi SQL Injection (**RNF3**).
Sicurezza utenti	Protezione dei dati personali degli utenti e delle transazioni di pagamento.
Sicurezza catalogo	Gli aggiornamenti al catalogo e alle scorte devono essere immediati (**RFN6**)

1.2.3. Manutenzione

Estendibilità	Facilità nell'aggiungere nuove funzionalità (come l'integrazione con nuovi gateway di pagamento).
Modificabilità	Possibilità di modificare funzionalità esistenti senza influire negativamente sul sistema.
Adattabilità	Capacità di adattarsi a diversi contesti o domini (ad es. un altro paese con diverse normative).
Leggibilità	Codice ben documentato e strutturato per facilitare la comprensione e il debug (**RNF10**).
Tracciabilità dei requisiti	Possibilità di mappare facilmente il codice alle specifiche originali del progetto.

1.2.4. Utente finale

Utilità	Il sistema deve supportare gli utenti nel completare i loro acquisti, gestire i carrelli, e seguire i processi di pagamento in modo semplice.
Usabilità	Il sistema deve essere intuitivo, con interfaccia user-friendly
Feedback	Il sistema deve fornire messaggi di conferma e notifiche degli errori in tempo reale (**RNF4**).
Accessibilità	Deve essere accessibile su vari dispositivi, come desktop, tablet e smartphone.

1.2.5. Compromessi

Spazio vs. Velocità	Potrebbe essere necessario utilizzare più memoria per ottenere tempi di risposta più rapidi
Tempo di consegna vs. Funzionalità	Ridurre le funzionalità per rispettare le scadenze di lancio del sito
Sicurezza vs. Usabilità	Aumentare la sicurezza (ad esempio, richiedendo password complesse e autenticazione a due fattori) può rendere il sistema meno intuitivo e creare frustrazione per gli utenti che cercano un'esperienza rapida e fluida.

1.3. Definizioni, acronimi e abbreviazioni

RNF: Requisito Non Funzionale
RAD: Requirements Analysis Document

1.4. Riferimenti

Nel corso del documento facciamo riferimento ai Requisiti Non Funzionali, agli Use Cases e al Class Diagram presenti nel documento RAD.

1.5. Panoramica

Nei paragrafi successivi presentiamo l'analisi architetturale della piattaforma BirdComics. La sua decomposizione in sottosistemi, strategie di deployment su hardware, strategie per la gestione dei dati persistenti ed eventuali Boundary Condition del sistema.

2. Architettura attuale

Il sistema che andremo a sviluppare rappresenta un progetto completamente nuovo e innovativo, poiché al momento non esistono soluzioni simili o analoghe nel mercato. Attualmente, non esiste una piattaforma dedicata alla vendita online di fumetti con le caratteristiche e le funzionalità che intendiamo implementare. Pertanto, il nostro progetto avrà una dimensione pionieristica, occupando uno spazio di mercato vuoto e rispondendo a esigenze non ancora adeguatamente soddisfatte da altri attori del settore.

3. Architettura software proposta

3.1. Decomposizione del sottoproblema

Il sistema che andremo a sviluppare sarà basato sullo stile architetturale MVC. Quindi avremo 3 sottoproblemi di media complessità dove per ognuno andiamo a decomporlo in altri sottoproblemi non ulteriormente decomponibili.
I principali sottoproblemi si concentrano sulla visione dello stile architetturale, suddivisa nei seguenti livelli:

- **Model**: all'interno di questo sottoproblema abbiamo tutti i meccanismi atti a gestire la logica del sistema, in quanto conoscono il dominio della soluzione. Include i meccanismi per il controllo delle informazioni e l'interfacciamento con il database implementati tramite classi wrapper. Questa scelta consente di ridurre l'accoppiamento e facilita eventuali futuri cambiamenti del DBMS. I sottosistemi che appartengono alla logica del sistema riguardano la gestione del:
    - Accessi: operazioni che si occupano di far autenticare gli attori all'interno del sito
    - Profili: operazioni che si occupano di far modificare le informazioni degli utenti connessi al sito
    - Carrello: operazioni riguardanti la gestione del carrello
    - Finanza: operazioni riguardo le fatturazioni aziendali
    - Catalogo: operazioni per gestire i fumetti a catalogo
    - Magazzino: operazioni per gestire la predisposizione degli scaffali

        - Spedizione: operazioni per gestire l’interfacciamento con il corriere permettendo al cliente di vedere lo stato della spedizione
        - Assistenza: operazioni per poter fornire supporto al cliente
        - ConnessioneDBMS: Classe wrapper per ridurre l’accoppiamento
        - OperazioniDBMS: Operazioni che implementano la classe wrapper, queste operazioni accedono direttamente al database prendendo o salvando dati persistenti.
- **View**: all’interno di questo problema abbiamo tutti gli oggetti che offrono l’interfaccia grafica agli attori che andranno a visitare il sito. Sono le classi che forniscono HTML al browser dell’utente, le pagine sono:
    - HomePage
    - Fumetto
    - Carrello
    - GestioneCatalogo
    - GestioneMagazzino
    - Assistenza
    - Finanza
    - Visualizzazione ordini da parte del cliente
    - Login
    - Registrazione
- **Control**: è responsabile della gestione delle interazioni degli utenti con l’interfaccia grafica. Queste interazioni vengono tradotte in richieste per il livello Model attraverso meccanismi progettati per ridurre l’accoppiamento tra le classi di controllo e quelle di gestione della logica. Sono le classi JSP ed AJAX che si occupano di gestire l’input dell’attore sulla pagina HTML, elaborare tale richiesta e passarla al model

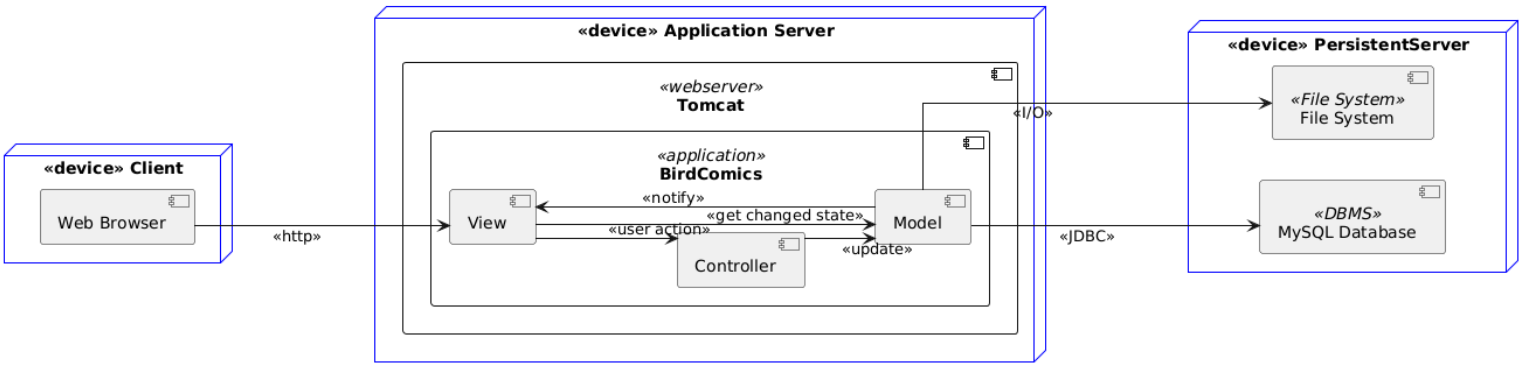
L'architettura MVC sarà implementata utilizzando il pattern di progettazione Observer per eliminare le dipendenze dirette tra il sottoproblema Model e il sottoproblema View. Grazie a questo approccio, combinato con meccanismi per ridurre l'accoppiamento tra Control e Model e l'uso di classi wrapper tra il Model e il DBMS, il sottoproblema Model risulta completamente isolato.

Questo isolamento garantisce una progettazione modulare, rendendo il sistema più facile da mantenere e aggiornare in futuro. Ad esempio, sarà possibile modificare l'interfaccia grafica o sostituire il DBMS senza impattare la logica applicativa, preservando così la stabilità del sistema e riducendo i costi di manutenzione.

3.2. Mappatura hardware/software

Gli utenti accedono all'applicazione BirdComics tramite un web browser sulla propria macchina client, stabilendo una connessione HTTP con il server. L'applicazione è ospitata nel web container Tomcat, che gestisce la logica dell'applicazione suddivisa tra i componenti View, Controller e Model.

Il Model interagisce con il File System per gestire file esterni e con il MySQL Database tramite JDBC per la gestione dei dati persistenti. Ogni volta che un utente interagisce con la View, questa invia l'azione al Controller, che a sua volta può aggiornare il Model. Il Model aggiorna lo stato dell'applicazione e notifica i cambiamenti alla View, che riflette le modifiche all'utente.



3.3. Gestione dei dati persistenti

La gestione dei dati persistenti è fondamentale per garantire che informazioni critiche siano conservate e recuperabili anche in caso di riavvio del sistema o di eventi imprevisti. Nel contesto del progetto, è stato deciso di utilizzare una strategia di gestione della persistenza basata su un Database Relazionale (MySQL) per sfruttare la sua capacità di gestire dati complessi e assicurare scalabilità, integrità e sicurezza.All'interno del database verranno salvate le seguenti informazioni

- Utenti: tutti i dati personali riguardanti gli attori registrati sul sito. Nel caso dei gestori salviamo anche il ruolo che ricoprono
- magazzino: oltre alle informazioni come nome e indirizzo salviamo anche gli scaffali che contiene
- scaffale: memorizziamo tutti i fumetti che sono stati catalogati su quello scaffale
- indirizzo
- fatture
- ordine
- carrello
- fumetto
- catalogo

3.4. Controllo degli accessi e sicurezza

Attori:

- **Ospite**
  - *GestioneAccessi*:
    - Registrazione
  - *GestioneCarrello:*
    - Aggiunta/Rimozione fumetto al carrello, Visualizza carrello, Svuota carrello, Modifica della quantità di un prodotto nel carrello
- **Cliente**
  - *GestioneAccessi*:
    - Autenticazione, Disconnessione
  - *GestioneProfili*:
    - Visualizzazione, modifica e annullamento delle modifiche al profilo
  - *GestioneCarrello:*
    - Aggiunta/Rimozione fumetto al carrello, Visualizza carrello, Svuota carrello, Modifica della quantità di un prodotto nel carrello
- **Direttore Generale**
  - *GestioneAccessi*:
    - Autenticazione, Disconnessione, Registrazione di un direttore magazzino
  - *GestioneProfili*:
    - Visualizzazione e modifica del proprio profilo e modifica dei profili direttore magazzino
- **Direttore Magazzino**
  - *GestioneAccessi*:
    - Autenticazione, Disconnessione, Registrazione di un dipendente con ruolo risorse umane
  - *GestioneProfili*:
    - Visualizzazione del proprio profilo e modifica dei profili di risorse umane
- **Risorse Umane**:
  - *GestioneAccessi*:
    - Autenticazione, Disconnessione, Registrazione di un dipendente con ruolo gestore
  - *GestioneProfili*:
    - Visualizzazione del proprio profilo e modifica dei profili dei ruoli di finanza, magazziniere, operatore logistico, gestore catalogo e assistenza
- **Finanza**
  - *GestioneAccessi*:
    - Autenticazione, Disconnessione
  - *GestioneProfili*:
    - Visualizzazione del proprio profilo
  - *GestioneFinanza*:
    - Visualizzare fatture di una data giornata

- **Magazziniere**
  - *GestioneAccessi*:
    - Autenticazione, Disconnessione
  - *GestioneProfili*:
    - Visualizzazione del proprio profilo
  - GestioneMagazzino
    - Aggiunta di un fumetto allo scaffale, Rimozione di un fumetto dallo scaffale

- **Operatore Logistico**
  - *GestioneAccessi*:
    - Autenticazione, Disconnessione
  - *GestioneProfili*:
    - Visualizzazione del proprio profilo
  - GestioneSpedizione:
    - Aggiunta/Modifica codice codice di tracciamento della spedizione

- **Gestore Catalogo**
  - *GestioneAccessi*:
    - Autenticazione, Disconnessione
  - *GestioneProfili*:
    - Visualizzazione del proprio profilo
  - GestioneCatalogo:
    - Aggiunta di un fumetto al catalogo, Rimozione di un fumetto dal catalogo, Aggiornamento di un fumetto del catalogo
- **Assistenza**
  - *GestioneAccessi*:
    - Autenticazione, Disconnessione
  - *GestioneProfili*:
    - Visualizzazione del proprio profilo
  - GestioneAssistenza:
    - Inserimento codice ordine, Visualizza ordine, Visualizza fattura

3.5. Controllo software globale

Il sistema BirdComics adotta un'architettura event-driven, in cui le richieste degli utenti vengono inizializzate tramite interazioni con l'interfaccia grafica. Quando un utente effettua un'azione, come cliccare su un fumetto per acquistarlo o aggiornare il proprio profilo, viene generato un evento. Per ogni evento generato dall'utente, esiste un rispettivo listener sul sistema che innesca un processo di elaborazione. L'elaborazione delle richieste avviene tramite un sistema distribuito, consentendo di gestire concorrenza, sincronizzazione e conflitto dei dati in modo efficace. Evitando che l'elaborazione di una richiesta blocchi altre operazioni di altri utenti connessi in contemporanea. Oltre al sistema distribuito, per garantire la consistenza dei dati, viene implementato un dbms mysql che si interfaccia con il sistema distribuito.

3.6. Condizioni boundary

Le condizioni boundary identificate per il nostro e-commerce includono situazioni legate all'avvio del sistema, il suo spegnimento, il riavvio in seguito a un fallimento.

Primo avvio

Prima del primo avvio, lo sviluppatore che sta configurando il sistema dovrà inserire manualmente almeno un profilo Direttore Generale all'interno del database MySQL, dopo aver avviato il relativo servizio. Questo è necessario perché, nella nostra piattaforma, solo un amministratore può registrare nuovi amministratori o gestori. Completata questa operazione, si potrà avviare il web server che ospita l'e-commerce.

Per gli avvii successivi, sarà sufficiente avviare il DBMS e il web server.

Avvio in caso di fallimento

In caso di un'interruzione improvvisa del sistema, il database utilizza transazioni per garantire la consistenza delle informazioni. Eventuali modifiche parziali in corso non verranno salvate, evitando inconsistenze nei dati.
Non sono previste funzionalità per il ripristino dello stato precedente all'arresto improvviso.

Spegnimento

Per spegnere correttamente il sistema, sarà sufficiente terminare l'esecuzione del web server. Poiché il web server gestisce anche la connessione al DBMS, non sarà necessario spegnere manualmente il database.
Eventuali transazioni non completate verranno annullate, garantendo la consistenza dei dati persistenti.

Errato accesso ai dati persistenti

Se si verificano errori nella configurazione del database o nell'utilizzo del driver per connettersi ad esso, il sistema mostrerà una pagina di errore personalizzata. Questa pagina fornirà dettagli sulla causa e sul punto preciso del codice sorgente in cui si è verificato il problema.
Lo sviluppatore dovrà correggere il problema prima di riavviare il sistema. Una volta risolto, il sistema tornerà operativo come di norma.