SYSTEM DESIGN DOCUMENT

“Peek A Book”

Immagine che contiene clipart, disegno, cartone animato, illustrazione

Descrizione generata automaticamente

|  |  |
| --- | --- |
| Versione | 0.1 |
| Data | 28/05/2024 |
| Presentato da | Iacomino Domenico, De Luca Ciro |

RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 28/05/2024 | 0.1 | Prima stesura | Iacomino Domenico,  De Luca Ciro |
| 29/05/2024 | 0.2 | Seconda Stesura | Iacomino Domenico,  De Luca Ciro |
|  |  |  |  |

Indice

Sommario

[1. Introduction 3](#_Toc167898169)

[1.1. Purpose of the system 3](#_Toc167898170)

[1.2. Design goals 3](#_Toc167898171)

[1.3. Definition, acronyms and abbreviations 5](#_Toc167898172)

[*1.4.* *References* 5](#_Toc167898173)

[*1.5.* *Overview* 5](#_Toc167898174)

[2. Current software architecture 6](#_Toc167898175)

[3. Proposed software architecture 7](#_Toc167898176)

[3.1. Overview 7](#_Toc167898177)

[3.2. Subsystem decomposition 7](#_Toc167898178)

[3.3. Hardware/Software Mapping 12](#_Toc167898179)

[3.4. Persistent data management 13](#_Toc167898180)

[3.5. Access control and security 16](#_Toc167898181)

[3.6. Global software control 18](#_Toc167898182)

[3.7. Boundary Conditions 19](#_Toc167898183)

1. Introduction
   1. Purpose of the system

Il gruppo Peek A Book intende sviluppare una applicazione web che mira ad essere il punto di riferimento per l’acquisto di libri in Italia ponendo l’accento sulla tempestività dell’approvvigionamento dei prodotti e sulla vicinanza al fruitore.

* 1. Design goals

In seguito all’analisi dei requisiti non-funzionali esplicitati nel RAD precedentemente consegnato e ad un’approfondita discussione con il committente, sono emersi i seguenti obiettivi di design ai quali il sistema software deve puntare in fase di sviluppo. Questi obiettivi sono organizzati nelle categorie che seguono:

**Performance:**

* Tempo di risposta: la schermata di visualizzazione dei prodotti deve essere visualizzata in al più 3 secondi.
* Throughput: il sistema deve garantire un carico minimo di 1000 utenti contemporaneamente connessi.
* Memoria: le immagini utilizzate per l’anteprima dei prodotti non devono superare i 300 KB in formato JPG e PNG.

**Dependability:**

* Robustezza: il sistema deve essere in grado di resistere ad ogni tipo di input da parte dell’utente validando le form compilate.
* Disponibilità: il sistema deve essere disponibile tutti i giorni 24h su 24.
* Fault tolerance: il sistema deve predisporre di pagine di errore per ogni classe di errore riscontrato.
* Security: il sistema deve garantire la resistenza agli attacchi, in particolar modo al database. Le password utente devono essere crittografate prima di essere salvate nel database.

**Maintenance:**

* Estensibilità: il sistema dovrebbe essere facilmente estensibile tramite l’aggiunta di nuove pagine.
* Adattabilità: il sito dovrebbe essere responsive e in grado di essere visualizzato sui browser più utilizzati.
* Portabilità: il sistema dovrebbe essere in grado di essere utilizzato su qualunque database relazionale e provider di server web.
* Tracciabilità dei requisiti: il sistema dovrebbe implementare ogni funzionalità in modo modulare.

**End User Criteria:**

* Usabilità: la struttura del sistema deve essere facilmente comprensibile alla visualizzazione.
* Utilità: il sistema dovrebbe garantire all’amministratore la possibilità di gestire facilmente le interazioni con il database tramite apposite pagine

Al fine di garantire la fattibilità di questi obiettivi nonché di sottolineare le priorità verso le quali deve rivolgersi lo sviluppo, sono stati di seguito specificati i compromessi da tenere in considerazione durante la produzione del sistema software:

* Se i tempi di rilascio sono stretti, può essere allentato il controllo sulle form compilabili soltanto dall’amministratore in favore di una gestione completa degli input delle form compilabili dall’utente.
* Se i tempi di rilascio sono stringenti, le funzionalità possono essere implementate secondo priorità a moduli.
* In caso di tempi brevi, sarebbe sempre da preferire l’aggiunta di nuove funzionalità a discapito della cura dell’estetica del sito. Potrebbe essere necessario spostare risorse umane dal team dedicato al front-end a quello dedicato al back-end.
* Potrebbe essere necessario comprimere le immagini relative ai prodotti per garantire un tempo di risposta accettabile.
  1. Definition, acronyms and abbreviations
* FK: chiave esterna, rappresenta un attributo relativo ad un’altra entità.
* Three-tier: un'architettura a tre livelli è un'architettura client-server in cui la logica di processo funzionale, l'accesso ai dati, l'archiviazione dei dati del computer e l'interfaccia utente sono sviluppati e mantenuti come moduli indipendenti su piattaforme separate.
  1. *References*

- B. Bruegge, A.H. Dutoit, Object Oriented Software Engineering – Using UML, Patterns and Java, Prentice Hall (3/E)

* 1. *Overview*

Le sezioni di questo documento sono articolate come segue:

* **Current software architecture:** si analizza l’architettura di sistema già disponibile,nel caso in cui non ci fosse nessun sistema pregresso (come nel nostro caso) allora si procede all’analisi dell’implementazione dei competitors.
* **Proposed software architecture:** si analizza l’implementazione del sistema nelle sue componenti e si commentano le scelte in merito all’architettura, al mapping hardware/software, alla persistenza e alla gestione degli accessi.
* **Subsystem services:** si analizza l’implementazione dei sottosistemi con particolare accortezza al loro ruolo.

1. Current software architecture

La struttura three-tier, è l'approccio di sviluppo software standard offerto dalla maggior parte dei framework Web più diffusi, è chiaramente un'architettura a strati. Ad interfacciarsi con il database si trova il Database layer, che contiene la logica di business e le informazioni sui tipi di dati nel database. Il Web layer invece rappresenta una visualizzazione delle componenti contenute nel sottosistema del Database layer. Essa è composta fondamentalmente da: file di stile CSS, JavaScript e HTML. A gestire il flusso del sistema c’è l’Application layer, che ha varie regole e metodi per trasformare i dati che si spostano tra il Web layer e il Database layer.

Il vantaggio di un'architettura a strati è la separazione dei compiti, ovvero ogni livello può concentrarsi esclusivamente sul proprio ruolo. Questo lo rende:

● Manutenibile

● Testabile

● Facile assegnare "ruoli" separati

● Facile da aggiornare e migliorare i livelli separatamente

Nonostante i vantaggi, il pattern Three-tier introduce anche un rallentamento nel flusso dei dati verso il database a causa dell’unica interfaccia con esso, l’Application layer, che si comporta da collo di bottiglia.

1. Proposed software architecture
   1. Overview

Le sezione del Proposed software architecture sono articolate come segue:

* **3.2 Subsystem decomposition:** analizziamo i vari sottosistemi del software e le loro principali interazioni
* **3.3 Hardware/Software mapping:** analizziamo l’interazione del software con l’hardware.
* **3.4 Persistent data management:** analizziamo come vengono archiviati i dati persistenti e l’architettura del database scelto.
* **3.5 Access control and security:** analizziamo come le entità possono accedere ad alcune parti del sistema.
* **3.6 Global software control:** analizziamo in che modo il sistema controlla il flusso degli eventi globale.
* **3.7 Boundary conditions:** analizziamo il comportamento del sistema nei suoi casi limite, ad esempio accensione e spegnimento.
  1. Subsystem decomposition

Il sistema divide le proprie componenti nei 3 sottosistemi principali citati dal pattern architetturale Three-tier.

Il sottosistema di Database layer, inoltre, si divide a sua volta in due ulteriori sottosistemi, uno dedicato alla rappresentazione di business degli oggetti persistenti (Entities) e un altro specializzato nel ruolo di interfaccia con il database (Storage).

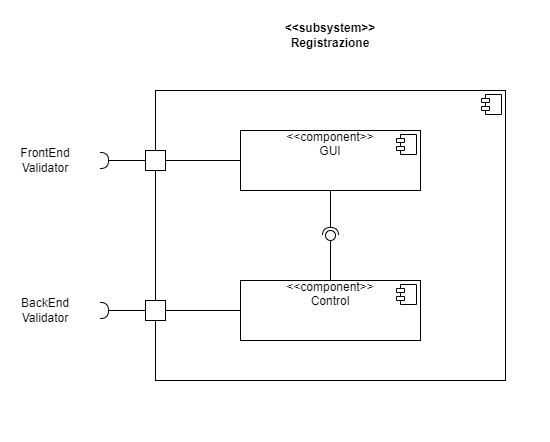
Il sottosistema Web layer, invece, contiene la visualizzazione dei dati secondo le diverse viste degli attori annoverati e tutti gli elementi boundary necessari all’interazione tra l’utente e il sistema. Esso è diviso in 6 package: Registrazione, Autenticazione, Catalogo, Carrello, Gestione Admin, Validator.

Il sottosistema Application layer si occupa della gestione delle richieste ed è diviso negli stessi package del Web layer. L’ Application layer gestisce il flusso di aggiornamento del database interfacciandosi con il sottosistema di Storage, imposta gli attributi degli oggetti del sottosistema Entities e inoltra al Client la pagina specifica contenuta nel sottosistema del Web layer.

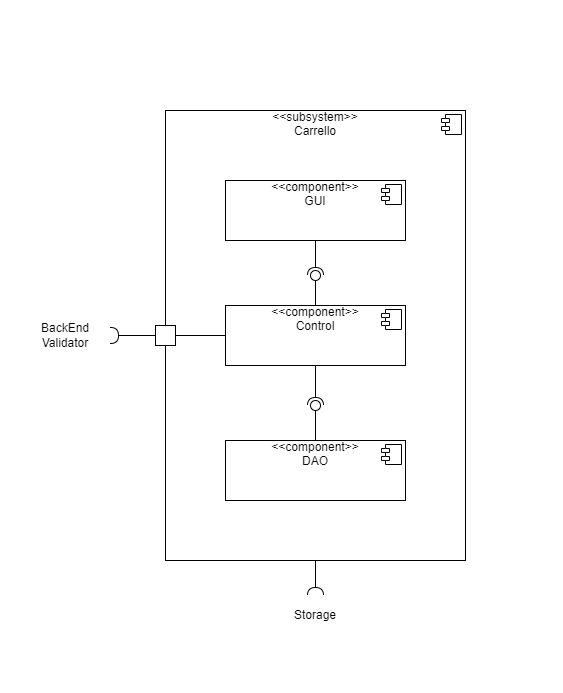
**Diagramma Architetturale**

Immagine che contiene testo, schermata, diagramma, Parallelo

Descrizione generata automaticamente

**Sottosistema Registrazione**

**Sottosistema Carrello**



**Sottosistema Autenticazione**

Immagine che contiene testo, diagramma, schermata, Parallelo

Descrizione generata automaticamente

Immagine che contiene testo, diagramma, Piano, Disegno tecnico

Descrizione generata automaticamente

**Sottosistema Gestione Admin**

Immagine che contiene testo, schermata, diagramma, Parallelo

Descrizione generata automaticamente**Sottosistema Catalogo**

* 1. Hardware/Software Mapping

A livello architetturale, il sistema software si pone l’obiettivo di essere mappato su 3 nodi fondamentali quali: Client, Web Server e Database Server.

Il nodo Client, in realtà, rappresenta la classe di nodi che si interfacciano dall’esterno all’e-commerce; tramite il Browser, il Client specifica l’URL del sito e apre una connessione http con il Web Server.

Le richieste inviate dal Client sono intercettate dal Web Server il quale gestisce il flusso del sistema e risponde inviando l’apposita pagina web. Nel caso di operazioni che coinvolgono i dati persistenti, il nodo Web Server invia le modifiche sotto forma di query al Database Server.

La connessione tra questi nodi è coordinata con protocollo HTTP e utilizza un connector per la gestione del collegamento con il database relazionale scelto.

Immagine che contiene diagramma, Disegno tecnico, schizzo, Piano

Descrizione generata automaticamente

* 1. Persistent data management

Per la gestione dei dati persistenti è stato scelto un database relazionale gestito tramite DBMS. Gli oggetti di tipo Entity saranno mappati in tabelle relazionali e gestiti dagli admin di sistema tramite interfaccia grafica in una sezione apposita del sito web. Come già specificato nel mapping hardware/software, la comunicazione tra il web server PeekABook e il database avverrà mediante il driver predisposto dalla tecnologia di archiviazione scelta(DBMS).

La scelta di una gestione tramite DBMS è stata considerata al fine di garantire:

* **Privatezza dei dati e diversità delle viste**, in quanto un DBMS garantisce un accesso protetto ai dati e una differente visualizzazione in base agli utenti.
* **Vincoli di integrità sui dati**, poiché un DBMS consente di specificare vincoli intrarelazionali e interrelazionali per mantenere l’integrità dei dati in seguito a modifiche della base dati.
* **Transazioni**, in quanto un DBMS gestisce in modo atomico una sequenza di operazioni tramite meccanismi di commit/rollback.

Rispetto al class diagram degli oggetti Entity specificato nella sezione Object Model del documento RAD, è stata valutata e accettata la scelta di mappare i due attori Utente e Amministratore accorpandoli in un’unica tabella.

Per discernere tra i due ruoli è stata introdotta nella tabella Utente la proprietà admin, un booleano che certifica il ruolo di amministratore in caso positivo.

Di seguito è specificato il diagramma ristrutturato delle entità che saranno mappate come tabelle nel database relazionale (cfr. RAD 3.4.3).

**Dizionario dei dati: Entità**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entità** | **Descrizione** | **Attributi** | **Identificatore** |
| Utente | Rappresenta un utente registrato al sistema. | Username, Email, Password, Nome, Cognome, Admin | username |
| Carrello | Mantiene nel database gli elementi aggiunti nel carrello di un utente | Totale, Numero elementi, Utente | utente  (FK Utente.username) |
| Ordine | Ordine effettuato da un utente | Numero Ordine, Numero elementi, Indirizzo Spedizione, Data, Metodo di pagmento | numero ordine |
| Prodotto | Rappresentazione del prodotto contenuto nel catalogo | Codice, Categoria, Nome, Prezzo, Copertina, Quantità, Descrizione | codice |

**Dizionario dei dati: Relazioni**

|  |  |  |  |
| --- | --- | --- | --- |
| **Relazione** | **Descrizione** | **Componenti** | **Attributi** |
| Possiede | Collegamento tra un utente e il suo carrello | Utente (1,1) Carrello (1,1) | - |
| Contiene | Descrive i prodotti contenuti nel carrello di un cliente | Carrello (0,\*) Prodotto (0,\*) | - |
| EffettuaOrdine | Descrive gli ordini effettuati da un cliente | Utente (1,1) Ordine (1,\*) | - |
| CompostoDa | Relazione che esplicita i prodotti presenti in un ordine effettuato | Ordine (0,\*) Prodotto (1,\*) | quantità acquistata |

* 1. Access control and security

Il sistema prevede due tipologie di utenti con ruoli diversi: l’admin e il cliente.

L’admin può accedere alla schermata di interazione con il database e alla schermata di visualizzazione delle statistiche del sito.

Il cliente può navigare tra i prodotti del catalogo ed effettuare ordini. Inoltre, può visualizzare le sue informazioni personali e gli ordini effettuati.

L’accesso al sito web è garantito mediante una form di login. I risultati della form saranno inviati al web server che cercherà l’accesso sul database e, in caso di corrispondenza, controllerà se per quell’utente è attivo il flag che indica il ruolo di admin. In caso positivo, il sistema reindirizzerà l’amministratore nella schermata di gestione del sito, altrimenti il cliente sarà riportato alla homepage con visualizzazione del profilo loggato.

**Matrice degli accessi**



* 1. Global software control

Il flusso di controllo globale è di tipo event-driven in quanto le richieste sono effettuate dall’utente tramite appositi comandi presenti nelle pagine del package Web layer.

Ad ogni comando è associata una richiesta HTTP che verrà intercettata dal web server e risolta tramite una classe specifica del package Application layer. Dopodiché, alla fine di ogni flusso, sarà invocato un dispatcher che provvederà a reindirizzare l’utente nella pagina corretta del package Web layer.

Le criticità sono rappresentate dall’interazione concorrente con il database, dall’accesso concorrente alla sessione da parte dell’utente e dall’accesso alla struttura dati condivisa che contiene i prodotti della homepage. In questi casi le richieste utente saranno sincronizzate e gestite sequenzialmente dal web server.

* 1. Boundary Conditions

**Prerequisiti:** per essere in funzione correttamente, il server deve avere a disposizione una connessione ad internet.

● **Startup:** quando un amministratore di sistema avvia il server tramite bottone apposito, il sistema si avvia e verifica se è possibile stabilire una connessione con il database per poter fornire tutte le funzionalità.

● **Shutdown:** quando un amministratore di sistema spegne il server tramite bottone apposito, il sistema controlla che non ci siano connessioni attive aperte con il database. In tal caso le chiude per preservare la consistenza dei dati.

● **Perdita di connessione con il database:** quando il sistema perde la connessione con il database, reindirizza le richieste su una pagina di errore specifica. Al ristabilirsi della connessione con il database, si effettua un backup e un controllo di consistenza delle tabelle.

**● Calo di tensione elettrica:** si attende il ritorno della connessione alla rete elettrica. Al ristabilirsi della connessione, si effettua un backup e un controllo di consistenza delle tabelle del database.

**Avvio del Sistema**



**Spegnimento del Sistema**



**Errore database**

