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 |
|  |  |  |  |
|  |  |  |  |

Indice

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**