OBJECT DESIGN

DOCUMENT

“Peek A Book”

Immagine che contiene clipart, disegno, cartone animato, illustrazione

Descrizione generata automaticamente

|  |  |
| --- | --- |
| Versione | 1.0 |
| Data | 03/07/2024 |
| Presentato da | Iacomino Domenico, De Luca Ciro |

RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 29/06/2024 | 0.1 | Prima Stesura | Iacomino Domenico,  De Luca Ciro |
| 02/07/2024 | 0.2 | Seconda stesura | Iacomino Domenico,  De Luca Ciro |
| 03/07/2024 | 1.0 | Terza stesura | Iacomino Domenico,  De Luca Ciro |

Indice

[1. Introduction 3](#_Toc170924265)

[1.1. Object design goals 3](#_Toc170924266)

[1.2. Object design trade-offs 3](#_Toc170924267)

[1.3. Tecnologie utilizzate 4](#_Toc170924268)

[2. Packages 5](#_Toc170924269)

[Package Autenticazione 6](#_Toc170924270)

[Package Carrello 7](#_Toc170924271)

[Package Catalogo 8](#_Toc170924272)

[Package Registrazione 9](#_Toc170924273)

[Package Storage 10](#_Toc170924274)

[Package Gestione\_Admin 11](#_Toc170924275)

[3. Class Interfaces 12](#_Toc170924276)

[ArticoloDAO 12](#_Toc170924277)

[AutoreDAO 17](#_Toc170924278)

[CarrelloDAO 19](#_Toc170924279)

[LibroDAO 20](#_Toc170924280)

[OrdineDAO 23](#_Toc170924281)

[UtenteDAO 26](#_Toc170924282)

[4. Class Diagram 31](#_Toc170924283)

[5. Design Pattern 35](#_Toc170924284)

1. Introduction

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. Object design goals

**Robustezza:** Il sistema deve essere in grado di gestire input imprevisti attraverso validazione, controllo degli errori e gestione delle eccezioni lanciate.

**Modificabilità ed Estensibilità:** La suddivisione in package deve garantire la modificabilità del sistema e la possibilità di aggiungere ulteriori funzionalità nelle fasi avanzate del rilascio.

* 1. Object design trade-offs

Come già specificato nella sezione Design goals nell’SDD, il sistema pone l’accento sulla modularità delle funzionalità da implementare(secondo la scala di priorità dei requisiti proposta all’interno del RAD), sulla responsività del sito web e sul controllo delle form inviate dall’utente(con particolare attenzione a quelle compilate dagli utenti non amministratori).

Al fine di adempiere a tali propositi e di intercettare i requisiti non funzionali, sono state effettuate le seguenti scelte:

* Salvataggio dell’utente nella sessione HTTP una volta loggato, per accelerare l’accesso ai dati personali. Ne risulta che la sessione HTTP deve prevedere un meccanismo di sincronizzazione.
* Gestione del carrello utente nella sessione HTTP una volta effettuato il login, in modo tale da aggiornare il carrello presente nel database soltanto una volta invalidata la sessione in seguito ad un logout. Ne risulta che la sessione HTTP deve prevedere un meccanismo di sincronizzazione.
* La validazione delle form viene effettuata frontend attraverso degli script in modo da escludere le form invalide dal flusso di richieste verso il server.
* Le immagini dei prodotti non potranno superare una grandezza massima di 1300x1600. Le immagini non potranno avere una risoluzione migliore al fine di diminuire l’occupazione di memoria secondaria del server e il tempo di caricamento delle pagine web.
  1. Tecnologie utilizzate

Il sistema sarà sviluppato utilizzando il linguaggio di programmazione Java con l’implementazione OpenJDK-17. Le classi che intercetteranno le richieste utente estenderanno la gerarchia HttpServlet e si occuperanno di gestire il flusso di controllo del sistema inviando pagine web coerenti come risposta.

Per la struttura delle pagine del sito si utilizzerà HTML5, si curerà l’aspetto estetico mediante l’utilizzo di stylesheet CSS3 e si renderanno interattive tramite script in JS.

Le pagine dinamiche, invece, saranno implementate mediante attraverso JSP. Per la responsiveness del sito web sono state utilizzate le media queries di CSS.

Il DBMS relazionale scelto per lo storing dei dati persistenti è MySQL. Il database sarà definito tramite script SQL e le connessioni ad esso saranno effettuate con l’utilizzo di jdbc/mysql-connector.jar.

Il deployment del sito avverrà in locale su un server HTTP attraverso Apache Tomcat v9.0.71.

1. Packages

In questa sezione verrà mostrato l’albero dei file del progetto in accordo ai packages individuati nel system design.

* + - * **src**, la cartella in cui sono contenuti tutti i file sorgente
        + **java**, contiene tutte le classi java e i suoi componenti
        + **test**, contiene tutte le classi di test
        + **webapp**, contiene tutti i file relativi alla parte front end
        + **target**, contiene il .war da deployare prodotto dalla build di Maven

Come esposto nell’SDD, il package principale è stato partizionato secondo il pattern architetturale MVC.

In particolare nel package Model troveremo le Entity e i DAO utilizzati per l’accesso al database.

La scelta del raggruppamento delle classi Entity è stata dettata dalle relazioni e

dipendenze individuate tra gli oggetti persistenti.

Immagine che contiene diagramma, testo, Piano, Rettangolo

Descrizione generata automaticamente

## **Package Autenticazione**

Immagine che contiene testo, schermata, Rettangolo, diagramma

Descrizione generata automaticamente

## **Package Carrello**

Immagine che contiene testo, schermata, Rettangolo, diagramma

Descrizione generata automaticamente

## **Package Catalogo**

Immagine che contiene testo, schermata, Parallelo, ricevuta

Descrizione generata automaticamente

## **Package Registrazione**

Immagine che contiene testo, schermata, Rettangolo, diagramma

Descrizione generata automaticamente

## **Package Storage**

Immagine che contiene testo, Parallelo, ricevuta

Descrizione generata automaticamente

## **Package Gestione\_Admin**

Immagine che contiene testo, Parallelo, schermata, ricevuta

Descrizione generata automaticamente

1. Class Interfaces

Di seguito saranno presentate le interfacce di ciascun DAO.

## **ArticoloDAO**





















## **AutoreDAO**









## **CarrelloDAO**







## **LibroDAO**













## **OrdineDAO**













## **UtenteDAO**

















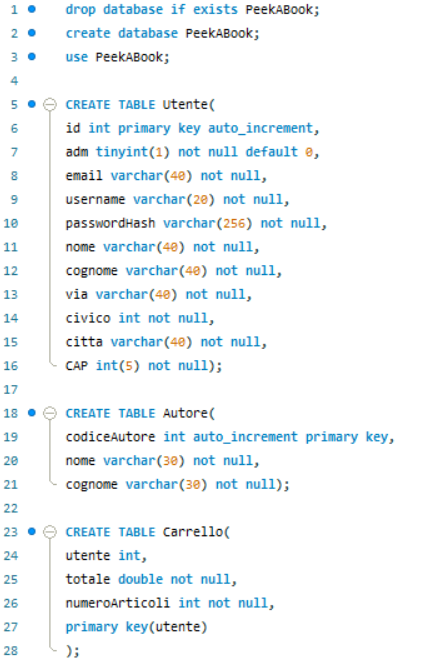
1. Class Diagram

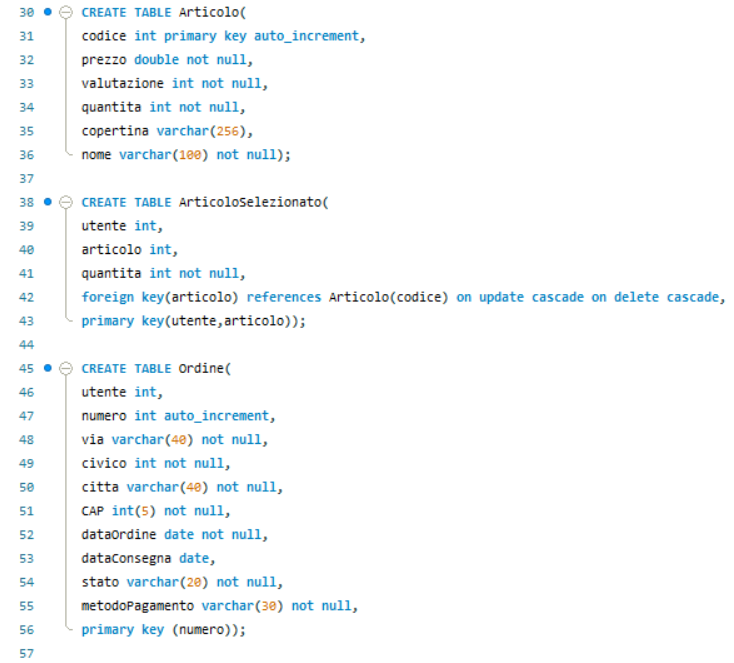
Si riporta di seguito il class diagram aggiornato relativo ai dati persistenti che il sistema PeekABook gestisce.

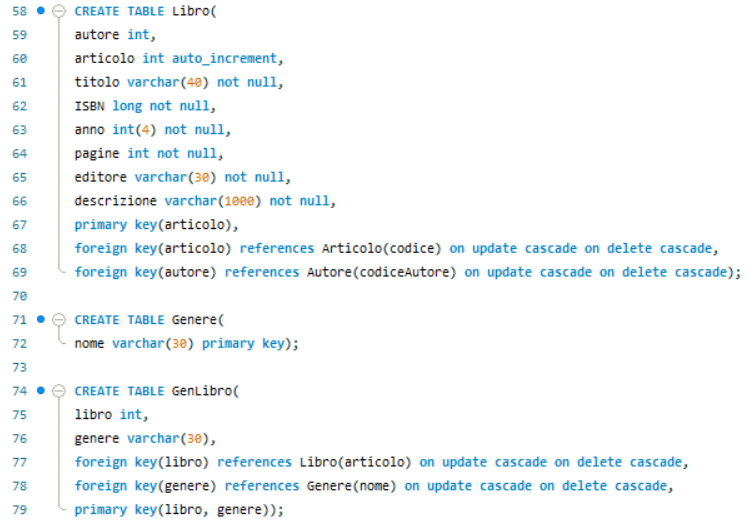
Immagine che contiene testo, diagramma, Piano, Parallelo

Descrizione generata automaticamente

La creazione del database, come espresso nella sezione 1.3 del documento corrente, è effettuata tramite il file di script peekDatabase.sql riportato di seguito.

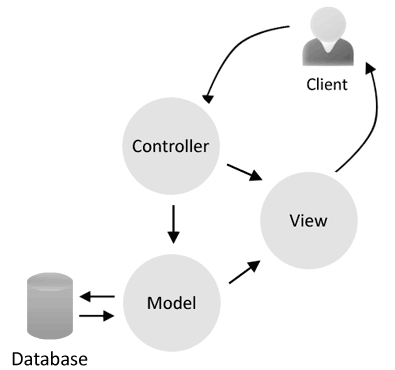






1. Design Pattern

Per lo sviluppo del progetto è stato utilizzato il pattern architetturale **MVC (Model-View-Controller)**. Esso permette di separare l’applicazione in tre componenti logici interconnessi ma allo stesso tempo indipendenti. Questa suddivisione aiuta a isolare in maniera chiara la logica di business, l’interfaccia utente, le interazioni utente e la gestione del dato rendendo il codice più facile da gestire e da estendere.



Inoltre è stato utilizzato il pattern **DAO** per separare la logica di business dalla logica di acceso ai dati. In questo modo, i componenti della logica di business non accedono mai direttamente al database. Solo gli oggetti previsti dal pattern Dao possono accedervi.  Inoltre, se dovessimo modificare il tipo di memoria persistente utilizzata, non sarà necessario stravolgere il codice della nostra applicazione, ma basterà modificare i DAO utilizzati.

Di seguito è riportato un esempio di utilizzo del pattern DAO in PeekABook.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente