**OBJECT DESIGN DOCUMENT**  
 VERSIONE 1.0 28/12/2020  
 2020/2021

**PROGETTO A CURA DI:  
  
COPPOLA PIETRO: 0512103934  
CARLO VITALE**: **0512103034**   


1° Versione: 29/10/2020

2° Versione: 10/11/2020

3° Versione: 24/11/2020

4° Versione:28/01/2021

1. Introduzione
2. Packages
3. Interfaccia delle Classi
4. Ottimizzazioni Class Diagram

**1. Introduzione  
  
1.1 Object Design trade-offs**Dopo aver stilato il documento di Requirements Analysis e il documento di System Design in cui vi è una descrizione sommaria di ciò che sarà il nostro sistema, definendo i nostri obiettivi ma tralasciando gli aspetti implementativi, andiamo ora a stilare il documento di Object Design che ha come obiettivo quello di produrre un modello che sia in grado di integrare in modo coerente e preciso tutte le funzionalità individuate nelle fasi precedenti.   
In particolar modo, in tale documento si definiscono le interfacce delle classi, le operazioni, i tipi, gli argomenti e la signature dei sottosistemi definiti nel System Design. Specificando inoltre i trade-off e le linee guida.

**Comprensibilità vs Tempo:** Il codice del sistema deve essere comprensibile il più possibile, in modo da facilitare la fase di testing ed eventuali future modifiche da apportare. Per rispettare queste linee guida il codice sarà accompagnato da commenti volti a semplificare la comprensione..

**Prestazioni vs Costi:** Dato che il nostro progetto è sprovvisto di budget, per poter mantenere prestazioni elevate, in determinate funzionalità verranno utilizzati dei template open source esterni, in particolare Bootstrap.

**Interfaccia vs Usabilità:** L’interfaccia grafica è stata realizzata in maniera molto semplice, chiara e concisa, vengono utilizzati i form e pulsanti con lo scopo di rendere semplice l’utilizzo del sistema da parte dell’utente finale.

**Sicurezza vs Efficienza:** La sicurezza rappresenta uno degli aspetti importanti del sistema. La sicurezza degli attori verrà implementata tramite controlli sui form, i quali potranno differenziare un semplice utente da un amministratore.

**1.2 Linee guida per la Documentazione delle interfacce**Gli sviluppatori dovranno seguire le seguenti linee guida per la stesura del codice:

**Naming Convention:**

È buona norma utilizzare nomi:

● Descrittivi

● Pronunciabili

● Di uso comune

● Di lunghezza medio-corta

● Non abbreviati

● Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

**Variabili:**

● I nomi delle variabili devono iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola. La dichiarazione delle variabili deve essere effettuata ad inizio blocco; in ogni riga vi deve essere una sola dichiarazione di variabile e va effettuato l’allineamento per migliorare la leggibilità. Esempio: idUtente, nomeUtente

● In determinati casi, è possibile utilizzare il carattere underscore “\_”, ad esempio quando si fa uso di variabili costanti oppure quando si fa uso di proprietà statiche.

**Metodi:**

● I nomi dei metodi devono iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola. Di solito il nome del metodo è costituito da un verbo che identifica un’azione, seguito dal nome di un oggetto. I nomi dei metodi per l’accesso e la modifica delle variabili dovranno essere del tipo getNomeVariabile() e setNomeVariabile(). Se viene dichiarata una variabile all’interno di un metodo quest’ultima deve essere dichiarata appena prima del suo utilizzo e deve servire per un solo scopo, per facilitare la leggibilità. Esempio: getId(), setId()  
  
**Classi e pagine:**

● I nomi delle classi e delle pagine devono iniziare con la lettera maiuscola, e anche le parole successive all’interno del nome devono iniziare con la lettera maiuscola. I nomi delle classi e delle pagine devono essere “evocativi” , in modo da fornire informazioni sullo scopo di quest’ultime.   
Esempio: Login.jsp  
  
● L’istruzione include che permette di importare all’interno della classe gli altri oggetti che la classe utilizza.

● La dichiarazione di una classe è caratterizzata da:

Dichiarazione della classe pubblica

Dichiarazioni di costanti

Dichiarazioni di variabili di classe

Dichiarazioni di variabili d’istanza

Costruttore

Commento e dichiarazione metodi e variabili

**1.3 Design Pattern**

**PATTERN DAO**

Pattern DAO viene utilizzato per separare i dati di basso livello che accedono alle API o alle operazioni, da servizi di alto livello.

Il vantaggio principale è il disaccoppiamento tra la business logic dell’applicazione e lo strato di persistenza dei dati (il database). Ciò permette un’architettura modulare, che semplifica la manutenzione del codice.

I partecipanti al pattern di accesso ai dati sono:

**Classe concreta dell'oggetto Accesso ai dati -** Questa classe è responsabile per ottenere dati da un'origine dati che può essere database / xml o qualsiasi altro meccanismo di archiviazione.

**Object Model** - questo oggetto è un semplice POJO che contiene i metodi get / set per archiviare i dati recuperati utilizzando la classe DAO.

**1.4 Definizioni, acronimi e abbreviazioni**

Acronimi:

● RAD: Requirements Analysis Document

● SDD: System Design Document

● ODD: Object Design Document

Abbreviazioni:

● DB: DataBase

Riferimenti:

● B. Bruegge, A. H. Dutoit, Object Oriented Software Engineering - Using UML, Pattern and Java, Prentice Hall, 3rd edition, 2009

**2. Packages**

Il sistema è suddiviso come segue:

**Package bean (.java) :**

|  |  |
| --- | --- |
| **CLASSE** | **DESCRIZIONE** |
| Carrello.java | Classe che rappresenta il carrello |
| Composizione.java | Classe che rappresenta la composizione di un ordine |
| Dettagli ordine | Classe che rappresenta un ordine completo di tutti i dettagli |
| DriverManagerConnectionPool.java | Classe di connessione al Database |
| Effettuato.java | Classe che rappresenta un ordine effettuato |
| Illustrazione.java | Classe che rappresenta le illustrazioni, cioè le immagini dei prodototti |
| IndirizzoOrdine.java | Classe che rappresenta l’indirizzo relativo ad un ordine |
| IndirizzoUtente.java | Classe che rappresenta l’indirizzo di un utente (che potrebbe non corrispondere a quello di spedizione di ogni ordine) |
| Ordine.java | Classe che rappresenta un ordine |
| Prodotto.java | Classe che rappresenta un prodotto |
| ProdottoAcquistato.java | Classe che rappresenta i prodotti acquistati che verranno visualizzati nel dettaglio di un ordine |
| ProdottoNelCarrello.java | Classe che rappresenta un prodotto inserito nel carrello |
| Utente.java | Classe che rappresenta un utente |

**Package Views (.jsp):**

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| acquistoRiuscito | Classe che visualizza un acquisto riuscito |
| areaAmministratore | Classe che visualizza la sezione dedicata agli amministratori del sito |
| avviso | Classe che visualizza un avviso legale |
| cambioPassword | Classe che visualizza un form per cambiare la password del proprio account |
| carrello | Classe che visualizza il carrello |
| dettagliOrdine | Classe che visualizza i dettagli di un Ordine |
| elenco | Classe che visualizza l’elenco di prodotti e relative illustrazioni |
| faq | Classe che visualizza le Frequents Answered Questions |
| fattura | Classe che visualizza la fattura di un ordine |
| home | Classe che visualizza la home page del sito |
| listaOrdiniUtente | Classe che visualizza una lista di ordini di un utente specifico |
| metodiPagamento | Classe che visualizza i metodi di pagamento consentiti sul sito |
| pagamento | Classe che visualizza il form per effettuare un pagamento |
| prodotto | Classe che visualizza un prodotto |
| registrazione | Classe che visualizza il form di registrazione al sito |
| spedizione | Classe che visualizza un form per inserire le specifiche di spedizione |
| termini | Classe che visualizza i termini d’uso del sito |
| utenteNonAutorizzato | Classe che visualizza un messaggio di mancata autorizzazione |
| visualizzaOrdiniAmministratore | Classe che visualizza gli ordini effettuati da un amministratore |

**Package Style (.css)**

|  |  |
| --- | --- |
| **Classe** | **Descrizionw** |
| style | Classe che contiene le direttive CSS |

**Package Elements (.jsp)**

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| aggiungiProdotto | Classe che visualizza un form di inserimento di un nuovo prodotto |
| eliminaPordotto | Classe che visualizza un form di cancellazione di un prodotto |
| footer | Classe che visualizza il footer delle pagine |
| header | Classe che visualizza l’header delle pagine |
| imageGallery | Classe che visualizza le gallerie con le immagini caricate |
| login | Classe che visualizza il form di login |
| modificaProdotto | Classe che visualizza il form di modifica di un prodotto |
| navbar | Classe che visualizza una barra di navigazione |
| search | Classe che visualizza una barra di ricerca testuale |

**Package Dao (.java)**

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| ComposizioneDao |  |
| EffetuatoDao |  |
| IlluastrazioneDao |  |
| IndirizzoOrdineDao |  |
| IndirizzoUtenteDao |  |
| OrdineDao |  |
| ProdottoDao |  |
| UtenteDao |  |

**3. Interfaccia Delle Classi**

|  |  |
| --- | --- |
| **Nome classe** | **Acquisto** |
| Descrizione | Rappresenta la classe che conferma un acquisto effettuato da un utente |
| Pre-condizione | **context** Acquisto:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: codiceFattura!=null && dataAcquisto!=null && utente!=null && carrello!= null && prodottiNelCarrello != null && indirizzoOrdine != null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **CambioPassword** |
| Descrizione | Rappresenta la classe che modifica la password di un account |
| Pre-condizione | **context** CambioPassword:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: utente!=null && passwordAttualeInserita!=null && nuovaPassword != null && passwordAttuale == passwordAttualeInserita |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **DettaglioOrdine** |
| Descrizione | Rappresenta la classe che permette di visualizzare il dettaglio di un ordine effettuato sul sito |
| Pre-condizione | **context** DettaglioOrdine:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: codiceFattura!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **EliminazioneProdotto** |
| Descrizione | Rappresenta la classe che elimina un prodotto dal sito in modo che non possa più essere visualizzato. |
| Pre-condizione | **context** EliminazioneProdotto:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: utente.isAmministratore==true && idProdotto!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **GenerazioneCatalogo** |
| Descrizione | Rappresenta la classe che elenca i prodotti hanno un catalogo specificato |
| Pre-condizione | **context** GenerazioneCatalogo:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: nomeCatalogo!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **GenerazioneFattura** |
| Descrizione | Rappresenta la classe che recupera e invia i dati che riguardano la fattura visualizzata prima di confermare l’acquisto |
| Pre-condizione | **context** GenerazioneFattura:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: utente!=null && carrello!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **GenerazionePaginaCarrello** |
| Descrizione | Rappresenta la classe che recupera e invia i dati del carrello e li invia al front-end per poterli visualizzare. |
| Pre-condizione | **context** GenerazionePaginaCarrello:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: utente!=null && carrello!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **GenerazionePaginaProdotto** |
| Descrizione | Rappresenta la classe che recupera e invia i dati di un prodotto e li invia al front-end per poterli visualizzare. |
| Pre-condizione | **context** GenerazionePaginaProdotto:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: idProdotto!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **InserimentoNelCarrello** |
| Descrizione | Rappresenta la classe che inserisce un prodotto nel carrello |
| Pre-condizione | **context** GenerazionePaginaProdotto:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: prodotto!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **InserimentoProdotto** |
| Descrizione | Rappresenta la classe che inserisce un prodotto nel sito |
| Pre-condizione | **context** InserimentoProdotto:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: nome!=null && modello!=null && catalogo!=null && materiale!=null && peso!=null && prezzo!=null && descrizione!=null && immagine!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **ListaOrdiniAmministratore** |
| Descrizione | Rappresenta la classe che carica tutti gli ordini effettuati sul sito per poterli far visionare agli amministratori. |
| Pre-condizione | **context** ListaOrdiniAmministratore:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: utente.isAmministratore==true |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **ListaOrdiniUtente** |
| Descrizione | Rappresenta la classe che carica tutti gli ordini effettuati su silto da un utente. |
| Pre-condizione | **context** ListaOrdiniUtente:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: utente!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **Login** |
| Descrizione | Rappresenta la classe che fa accedere un utente al sito |
| Pre-condizione | **context** Login:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: email!=null && password!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **Login** |
| Descrizione | Rappresenta la classe che fa accedere un utente al sito |
| Pre-condizione | **context** Login:: doGet(HttpServletRequest request, HttpServletResponse response) **pre**: email!=null && password!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **Logout** |
| Descrizione | Rappresenta la classe che disconnette l’account di un utente dal sito |
| Pre-condizione | **context** Logout:: doGet(HttpServletRequest request, HttpServletResponse response) **pre:** utente!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **ModificaProdotto** |
| Descrizione | Rappresenta la classe che permette a un amministratore di modificare un prodotto presente sul sito |
| Pre-condizione | **context** ModificaProdotto:: doGet(HttpServletRequest request, HttpServletResponse response) **pre:** utente.isAmministratore==true && idProdotto!=null && scelta modifica!=null && (nuovoPrezzo!=null || nuovoSconto!=null || nuovaQuantita!=null) |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **Registrazione** |
| Descrizione | Rappresenta la classe che registra un utente nel sito |
| Pre-condizione | **context** Registrazione:: doGet(HttpServletRequest request, HttpServletResponse response) **pre:** utente==null && nome!=null && cognome!=null && email!=null && password!=null && numeroTel!=null && città!=null && cap!=null && provincia!=null && indirizzo!=null && numCivico!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **Ricerca** |
| Descrizione | Rappresenta la classe che seleziona i risultati di una ricerca di un utente |
| Pre-condizione | **context** Ricerca:: doGet(HttpServletRequest request, HttpServletResponse response) **pre:** ricerca!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **RimozioneProdottoCarrello** |
| Descrizione | Rappresenta la classe che rimuove un prodotto dal carrello |
| Pre-condizione | **context** RimozioneProdottoCarrello:: doGet(HttpServletRequest request, HttpServletResponse response) **pre:** utente!=null && carrello!=null |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **Spedizione** |
| Descrizione | Rappresenta la classe che gestisce i dati di spedizione di un prodotto |
| Pre-condizione | **context** Spedizione:: doGet(HttpServletRequest request, HttpServletResponse response) **pre:** utente!=null && (indirizzoUtente!=null || (città!=null && indirizzo!=null && provincia!=null && numeroCivico!=null && cap!=null)) |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **Spedizione** |
| Descrizione | Rappresenta la classe che gestisce i dati di spedizione di un prodotto |
| Pre-condizione | **context** Spedizione:: doGet(HttpServletRequest request, HttpServletResponse response) **pre:** utente!=null && (indirizzoUtente!=null || (città!=null && indirizzo!=null && provincia!=null && numeroCivico!=null && cap!=null)) |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| **Nome classe** | **TizzoneArdenteContextListener** |
| Descrizione | Rappresenta la classe che inizializza alcune funzionalità del sistema |
| Pre-condizione |  |
| Post-condizione |  |
| Invarianti |  |