Object Design Document

## Introduzione

1.1 Object Design Trade-offs

Questa sezione descrive i compromessi generali fatti dagli sviluppatori. I trade-offs individuati sono i seguenti.

Funzionalità vs Usabilità: Trattandosi di un sito di e-commerce si preferisce dare priorità all’usabilità rispetto alla funzionalità

Affidabilità vs Costo: Nonostante risulti essere più dispendioso, si preferisce utilizzare meccanismi di controllo dell’inserimento dei dati in input al fine di evitare errori

Prestazioni vs Modificabilità: Trattandosi di una piattaforma web si preferisce dare priorità alla performance piuttosto che decomporre il sistema a grana fine

Buy vs Build: il nostro sistema si concretizza in una web-app, per la sua realizzazione faremo uso delle numerose librerie e framework, off-the-shelf, sfruttando anche i principi del riuso.

Le componenti off-the-shelf sono state selezionate sulla base sia delle funzionalità offerte da tali tecnologie, sia delle conoscenze pregresse dei membri del team al fine di evitare lo spreco di risorse temporali per familiarizzare con una nuova tecnologia sconosciuta.

*Memory Space vs Performance*: il sistema deve garantire un tempo di risposta relativamente basso, quindi se necessario verrà allocata più memoria al fine di diminuire i tempi di risposta.

1.1.2 Design Pattern

Per realizzare il riuso si è scelto di utilizzare soluzioni a problemi ricorrenti, queste soluzioni sono realizzate dai seguenti design patterns.

Nome: Observer Pattern

Descrizione del problema: necessità di notificare il Model, la View e il Control dato che tra loro esiste un forte legame in merito al cambiamento di stato.

Soluzione: Definire Subject e Observer

Nel legame View-Model il Subject è il Model, mentre l’Observer è la View.

Nel legame View-Controller il Subject è il Controller, mentre l’Observator è la View.

1.2 Linee guida per la documentazione dell’interfaccia

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Regola per la Nomenclatura** | **Esempi** |
| **Package** | I nomi dei package devono iniziare con una lettera maiuscola e tutte minuscole  I nomi dei package non devono contenere spazi. L’unico carattere speciale permesso è “\_” | View  Model  Control  Gestione\_Utente |
| **Classi** | Le classi sono denominate con nomi singolari.  Le classi relative ai Bean devono avere la nomenclatura: EntitàBean  Le classi relative ai Control devono avere la nomenclature: SostativoControl  Le classi relative ai DAO devono avere la nomenclatura: EntitàDAO  Le classi relative ai DTO devono avere la nomenclatura: EntitàDTO  Le interfacce relative ai DAO devono avere la nomenclatura: EntitàDAOIn | ProdottoBean  OrdineControl  ProdottoDAO  ProdottoDTO  ProdottoDAOIn |
| **JSP** | I documenti JSP devono avere la nomenclatura: SostantivoView | LoginView |
| **Metodi** | I metodi sono denominati con frasi verbali, campi e parametri con frasi nominali. | saveOrdine() |
| **Eccezioni** | Ogni eccezione deve avere nome esplicativo del problema segnalato | ProdottoNonDisponibile |

1.3 Definizioni, acronimi e abbreviazioni

N/A

1.4 Riferimenti

Requisiti non funzionali: 2.3 del RAD

## Pacchetti

2.1 Divisione in pacchetti

Il sistema sarà suddiviso in 3 package principali:

* Model
* View
* Control

L’utilizzo del pattern architetturale Model-View-Control è stato deciso nella fase di System Design.

|  |  |
| --- | --- |
| Main Package | Control |
| Descrizione | Contiene i package che si occupano del controllo della logica di business e al fine di produrre i risultati da inoltrare al Layer di Presentazione |
|  |  |
| Nome Package | Product\_Manager |
| Descrizione | Contiene le Servlet che permettono di: visualizzare, inserire, aggiornare e rimuovere un prodotto dal catalogo; visualizzare i prodotti in base al tipo, annata e categoria. |
| Dipendenze | BEANS |
|  |  |
| Nome Package | Order\_Manager |
| Descrizione | Contiene le Servlet che permettono la visualizzazione degli ordini, dello storico, del dettaglio ordine e della fattura |
| Dipendenze | BEANS |
|  |  |
| Nome Package | User\_Manager |
| Descrizione | Contiene le Servlet che permettono la registrazione di utenti e gestori, permettono di gestire il login e il logout e la gestione dei dati del cliente |
| Dipendenze | BEANS |
|  |  |

|  |  |
| --- | --- |
| Main Package | Model |
| Descrizione | Contiene i package che gestiscono i dati persistenti e la logica di business |
|  |  |
| Nome Package | DAO |
| Descrizione | Contiene le classi che gestiscono i dati persistenti, le interfacce e i DTO |
| Dipendenze | N/A |
|  |  |
| Nome Package | BEANS |
| Descrizione | Contiene i beans che si occupano della logica di business |
| Dipendenze | DAO |

|  |  |
| --- | --- |
| Main Package | View |
| Descrizione | Contiene i package che si occupano dell’interazione attraverso componenti grafiche che rappresentano le funzionalità del sistema |
|  |  |
| Nome Package | Catalogo |
| Descrizione | Contiene le JSP che raggruppa l’interfaccia grafica per la visualizzazione e la ricerca dei prodotti e per la gestione del catalogo |
| Dipendenze | Product\_Manager |
|  |  |
| Nome Package | Carrello |
| Descrizione | Contiene le JSP che raggruppa l’interfaccia grafica per l’inserimento e la gestione dei prodotti nel carrello |
| Dipendenze | Product\_Manager |
|  |  |
| Nome Package | Ordine |
| Descrizione | Contiene le JSP che raggruppa l’interfaccia grafica per la visualizzazione degli ordini e per la gestione della lista ordini |
| Dipendenza | Order\_Manager |
|  |  |
| Nome Package | Profilo\_Utente |
| Descrizione | Contiene le JSP che raggruppa l’interfaccia grafica per gestire i dati del cliente |
| Dipendenze | User\_Manager |
|  |  |
| Nome Package | Login\_Logout |
| Descrizione | Contiene le JSP che raggruppa l’interfaccia grafica per: login/logout, recupero password, registrazione di un cliente o gestore |
| Dipendenze | User\_Manager |

2.2 Organizzazione del codice in file

## Interfacce delle classi

## Class Diagram