****

Object Design Document

**Versione <1.00>**

****

Sommario

[1. Introduzione 3](#_Toc532204997)

[**1.1 Object design trade-offs** 3](#_Toc532204998)

[***1.1.2 Prestazioni vs Costi*** 3](#_Toc532204999)

[***1.1.3 Costi vs Mantenimento*** 3](#_Toc532205000)

[***1.1.4 Interfaccia vs Easy-use*** 3](#_Toc532205001)

[***1.1.6 Sicurezza vs costi*** 3](#_Toc532205002)

[***1.1.7 Interfacce vs Tempo di risposta*** 3](#_Toc532205003)

[**1.2 Descrizione componenti off-the-shelf** 3](#_Toc532205004)

[**1.3 Linee guida per la documentazione dell’interfaccia** 3](#_Toc532205005)

[**1.4 Definizioni, acronimi e abbreviazioni** 4](#_Toc532205006)

[**1.5 Riferimenti** 5](#_Toc532205007)

[3. Packages 6](#_Toc532205008)

[**PK1 - Package General** 6](#_Toc532205009)

[**PK2 - Package Interface Generale** 7](#_Toc532205010)

[***PK2.1 - Package Interface GestioneUtente*** 8](#_Toc532205011)

[***PK2.2 - Package Interface GestioneAdmin*** 8](#_Toc532205012)

[***PK2.3 - Package Interface GestioneProdotto*** 9](#_Toc532205013)

[***PK2.4 - Package Interface per la Gestione degli Errori*** 9](#_Toc532205014)

[**PK3 - Entity** 10](#_Toc532205015)

[***Utente*** 10](#_Toc532205016)

[***Admin*** 11](#_Toc532205017)

[***Ordine*** 11](#_Toc532205018)

[***Prodotto*** 12](#_Toc532205019)

[**PK4 – Gestioni** 13](#_Toc532205020)

[***Gestione Utente*** 14](#_Toc532205021)

[***Gestione Admin*** 14](#_Toc532205022)

[***Gestione Prodotto*** 15](#_Toc532205023)

[***Gestione Errori*** 15](#_Toc532205024)

[**PK5 – Storage** 16](#_Toc532205025)

[***Database*** 16](#_Toc532205026)

[***DatabaseQuery*** 16](#_Toc532205027)

# 1. Introduzione

## **1.1 Object design trade-offs**

Dopo una prima analisi, si sono osservati quali trade-off sono stati scelti per l’implementazione del sistema:

### ***1.1.2 Prestazioni vs Costi***

Il sistema è stato sviluppato utilizzando componenti *free* e *open source,* in cui non sono garantite prestazioni alte, saranno comunque soddisfacenti per il normale utilizzo, in particolare è stato scelto il cloud web services free offerto da Microsft Azure per l’Application Layer e il servizio che ospita lo storage Layer free: GearHost.

### ***1.1.3 Costi vs Mantenimento***

Il sistema è assente di qualsiasi documentazione o commento.

### ***1.1.4 Interfaccia vs Easy-use***

L’interfaccia, grazie ad una impostazione semplice e intuitiva, è di facile utilizzo (Easy-use) per le principali funzionalità del sistema anche per gli utenti meno esperti.

### ***1.1.6 Sicurezza vs costi***

Non sono state utilizzate componenti esterne che garantiscano la massima sicurezza sui dati, ma è stata usata la sicurezza intrinseca linguaggio Java che permetta un grado di protezione soddisfacente.

### ***1.1.7 Interfacce vs Tempo di risposta***

Il tempo di risposta tra server e interfaccia è estremamente basso.

## **1.2 Descrizione componenti off-the-shelf**

Per il progetto software che si vuole realizzare facciamo uso di componenti***off-the-shelf****,* che sono componenti software disponibili sul mercato per facilitare la creazione del progetto.

Per il sistema realizzato sono stati utilizzati componenti JQuery in particolare: [ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.min.js](http://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.min.js).

## **1.3 Linee guida per la documentazione dell’interfaccia**

Per rendere il codice più comprensibile sono stati aggiunti dei commenti sopra ogni metodo, utili per una spiegazione basilare del comportamento della funzione, insufficienti, però, a garantirne la completa comprensione.

**Commenti**

I commenti di implementazione sono, di solito, di una riga di codice che spiega solo in maniera basilare cosa fa il metodo.

**Dichiarazioni**

Le dichiarazioni sono posizionate a volte all’inizio dei metodi, a volte nel corpo del metodo, questa modalità rende il codice poco manutenibile e di difficile comprensione.

**Indentazione**

L’indentazione è stata effettuata con l’indentazione automatica dell’IDE utilizzata (Eclipse) e a prescindere dal linguaggio usato per la produzione del codice. Ci sono però punti in cui il codice non è stato indentato correttamente.

*Es.*:

<html>

<head>

</head>

<body>

</body>

</html>

**Parentesi**

A prescindere dalle istruzioni che seguono un IF o ciclo FOR e WHILE, è stata usata, laddove ci fosse anche una sola istruzione, il blocco di istruzioni tra parentesi graffe.

**Script Javascript**

Gli script che svolgono funzioni distinte dal funzionamento di una pagina, sono stati collocati in file separati.

Le funzioni e oggetti Javascript non sono stati preceduti da un commento in stile Javadoc, e sono di difficile individuazione.

## **1.4 Definizioni, acronimi e abbreviazioni**

**HTML:** Linguaggio di mark-up per pagine web  
**CSS:** Linguaggio usato per definire la formattazione di pagine web  
**Framework:** Software di supporto allo sviluppo web.

**Javascript:** [Linguaggio di scripting](https://it.wikipedia.org/wiki/Linguaggio_di_scripting) [orientato agli oggetti](https://it.wikipedia.org/wiki/Programmazione_orientata_agli_oggetti) e agli [eventi](https://it.wikipedia.org/wiki/Programmazione_orientata_agli_eventi), comunemente utilizzato nella [programmazione Web](https://it.wikipedia.org/wiki/Programmazione_Web) [lato client](https://it.wikipedia.org/wiki/Lato_client) per la creazione, in [siti web](https://it.wikipedia.org/wiki/Sito_web) e [applicazioni web](https://it.wikipedia.org/wiki/Applicazione_web).

**Ecplise:** ambiente di sviluppo integrato multi-linguaggio e multi-piattaforma

**Javadoc:** è un applicativo incluso nel Java Development Kit utilizzato per la generazione automatica della documentazione del codice sorgente scritto in linguaggio Java.

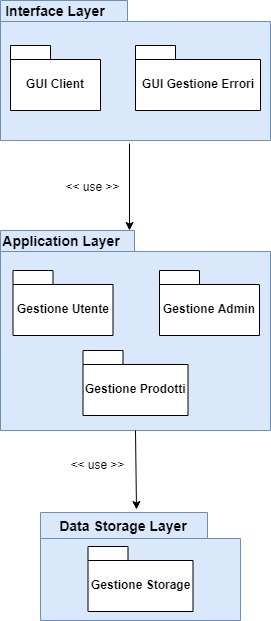
**Off-The-Shelf:** Servizi esterni di cui viene fatto utilizzo da terzi.

## **1.5 Riferimenti**

* Bernd Bruegge & Allen H. Dutoit, *Object-Oriented Software Engineering: Using UML, Patterns and Java*, (2nd edition),Prentice-Hall, 2003
* Ian Sommerville, Software Engineering, Addison Wesely

# 3. Packages

## **PK1 - Package General**



Il diagramma descrive la natura three-layer dell’applicazione mostrandone i tre package principali:

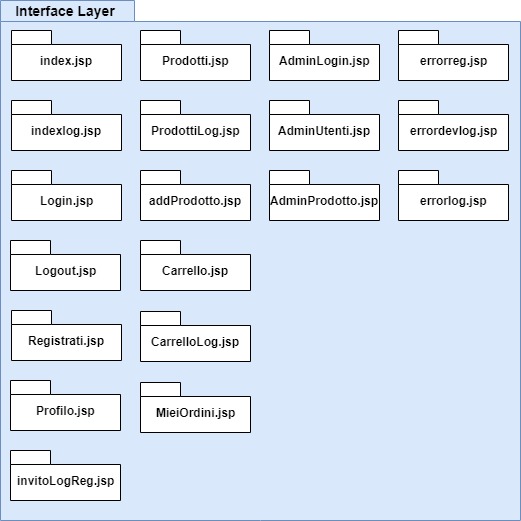
**InterfaceLayer: GUIClient e GUIGestioneErrori,** indicano i sottosistemi che contengono tutti gli oggetti boundary;

**ApplicationLogicLayer:** contiene i quattro sottosistemi individuati (Gestione Utente, Gestione Vendite, Gestione Aule Libere e Gestione Materiale Didattico) e il Gestore Errori;

**DataStorageLayer: GestoreStorage,** sottosistema che ha il compito di effettuare operazioni verso il database.

Il package Gestione Utente è un package indispensabile per il corretto funzionamento di alcune funzionalità dei package Gestione Vendite e Gestione Materiale Didattico. In particolare per alcune funzionalità sarà necessario assicurare un controllo sul package GU primi di accedere a pieno alle funzionalità del sistema USU.

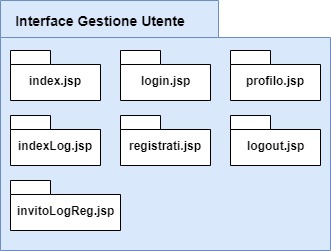
## **PK2 - Package Interface Generale**



Il diagramma descrive le interfacce delle varie sezioni del sistema:

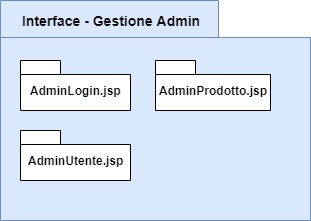
**Gestione Utente, Gestione Prodotti, Gestione Admin,** e le interfacce per la **Gestione degli Errori.**

### ***PK2.1 - Package Interface GestioneUtente***



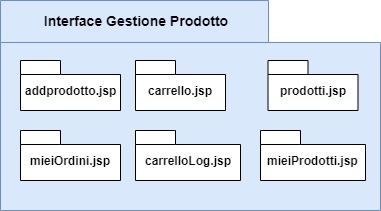
Il diagramma descrive le interfacce del sottosistema “Gestione Utente” ed in particolare le View per l’utente.

### ***PK2.2 - Package Interface GestioneAdmin***



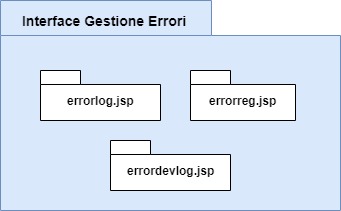
Il diagramma descrive le interfacce del sottosistema “Gestione Admin” ed in particolare le View per l’admin del sistema.

### ***PK2.3 - Package Interface GestioneProdotto***



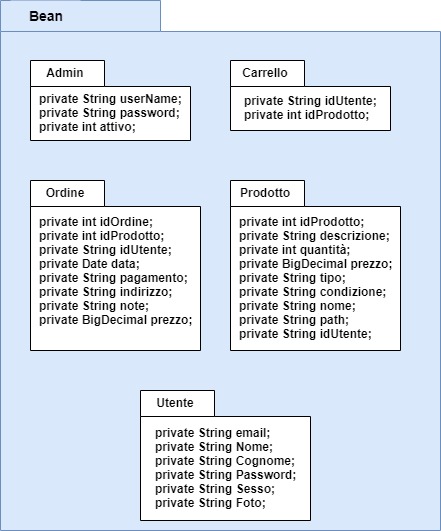
Il diagramma descrive le interfacce del sottosistema “Gestione Prodotto”.

### ***PK2.4 - Package Interface per la Gestione degli Errori***



Il diagramma descrive le interfacce del sottosistema “Gestione degli Errori”.

## **PK3 - Entity**



Il package principale “bean” contiene al suo interno i package delle entity del nostro sistema.

Per ogni entity sono definite le variabili e il tipo di queste.

### ***Utente***

|  |  |
| --- | --- |
| **Nome** | Utente |
| **Descrizione** | Questa classe rappresenta l’utente registrato al sito (Sottosistema Gestione Utente) |
| **Signature dei metodi** | getEmail()  setEmail(String email)  getNome()  setNome(String nome)  getCognome()  setCognome(String cognome)  getPassword()  setPassword(String password)  getSesso()  setSesso(String sesso)  getFoto()  setFoto(String foto) |
| **Precondizione** | **context:** Utente :: setEmail(email) **pre:** email non deve avere altre corrispondenze nel database. |
| **Post-condizione** |  |

### ***Admin***

|  |  |
| --- | --- |
| **Nome** | Admin |
| **Descrizione** | Questa classe rappresenta l’admin  (Sottosistema Gestione Admin) |
| **Signature dei metodi** | getUserName()  getUserName()  setUserName(String userName)  getPassword()  setPassword(String password  isAttivo()  setAttivo(int attivo)  toString()  setStatus(Boolean Status) |
| **Precondizione** | **context:** Utente :: setUserName(userName) **pre:** userName non deve avere altre corrispondenze nel database. |

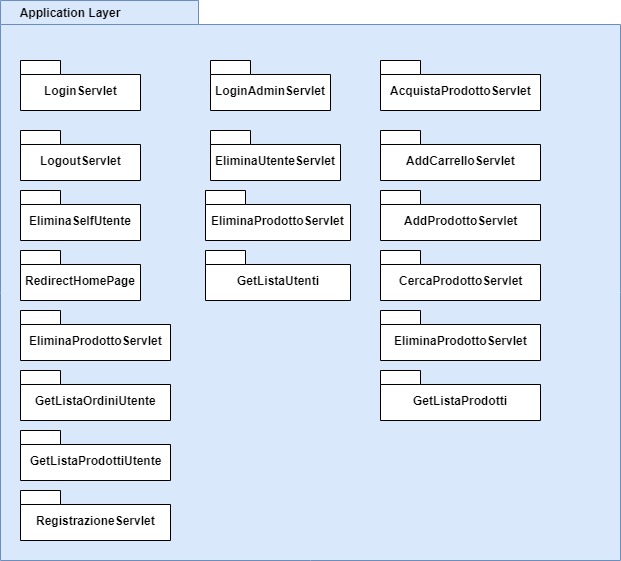
### ***Ordine***

|  |  |
| --- | --- |
| **Nome** | Ordine |
| **Descrizione** | Questa classe rappresenta gli ordini.  (Sottosistema Gestione Prodotti) |
| **Signature dei metodi** | getIdOrdine()  setIdOrdine(int idOrdine)  getIdProdotto()  setIdProdotto(int idProdotto)  getIdUtente()  setIdUtente(String idUtente)  getData()  setData(Date data)  getPagamento()  setPagamento(String pagamento)  getIndirizzo()  setIndirizzo(String indirizzo)  getNote()  setNote(String note)  getPrezzo()  setPrezzo(BigDecimal prezzo)  toString() |
| **Precondizione** | **context:** Utente: setIdOrdine(idOrdine) **pre:** L’utente che effettua un ordine deve essere loggato. |

### ***Prodotto***

|  |  |
| --- | --- |
| **Nome** | Prodotto |
| **Descrizione** | Questa classe definisce quelle che sono le caratteristiche di un prodotto, con le relative funzionalità ai quali essi sono soggetti. |
| **Signature dei metodi** | getIdProdotto()  setIdProdotto(int idProdotto)  getDescrizione()  setDescrizione(string descrizione)  getQuantità()  setQuantità(int quantità)  getPrezzo()  setPrezzo(BigDecimal prezzo)  getTipo()  setTipo(String tipo)  getCondizione()  setCondizione(String condizione)  getNome()  setNome(String nome)  getPath()  setPath(String path)  getUtente()  setIdUtente(String utente) |
| **Precondizione** |  |
| **Post-condizione** |  |

## **PK4 – Gestioni**



### ***Gestione Utente***

|  |  |
| --- | --- |
| **Nome** | Gestione Utente |
| **Descrizione** | Questa gestione si occupa dell’application layer ‘Gestione Utente’ |
| **Signature delle servlet** | LoginServletjava  LogoutServlet.java  EliminaSelfUtente.java  RedirectHomePage.java  EliminaProdottoServlet.java  GetListaOrdiniUtente.java  GetListaProdottiUtente.java  RegistrazioneServlet.java |
| **Precondizione** | **context:** EliminaSelfUtente.java:: **pre:** l’utente per eliminarsi deve essere loggato al sistema.  **context**: ElimaProdottoServlet.java:: **pre**: l’utente può eliminare solo i prodotti che sono stati inseriti da lui stesso.  **context**: GetListaOrdiniUtente.java **pre**: l’utente deve essere loggato per poter prendere visione dei suoi ordini.  **context**: GetListaProdottiUtente.java **pre**: l’utente deve essere loggato per poter prendere visione dei suoi prodotti. |
| **Post-condizione** |  |

### ***Gestione Admin***

|  |  |
| --- | --- |
| **Nome** | gestioneAdmin |
| **Descrizione** | Questa gestione si occupa dell’application layer di “Gestione Admin” |
| **Signature delle servlet** | LoginAdminServlet.java  EliminaUtenteServlet.java  EliminaProdottoServlet.java  GetListeUtenti.java |
| **Precondizione** |  |
| **Post-condizione** |  |
| **Invariante** |  |

### ***Gestione Prodotto***

|  |  |
| --- | --- |
| **Nome** | Gestione Prodotto |
| **Descrizione** | Questa gestione si occupa dell’application layer ‘Gestione Prodotto’ |
| **Signature delle servlet** | AcquistaProdottoServlet.java  AddCarrelloServlet.java  AddProdottoServlet.java  CercaProdottiServlet.java  EliminaProdottoServlet.java  GetListaProdotti.java |
| **Precondizione** | **context:** EliminaSelfUtente.java:: **pre:** l’utente per eliminarsi deve essere loggato al sistema. |
| **Post-condizione** |  |

### ***Gestione Errori***

La gestione degli Errori non è affidata a classi specifiche, ma gestita in ogni singola classe.

## **PK5 – Storage**

Il package storage è formato da quattro classi principali che si occuperanno di interfacciarsi al database e di passare le varie query.

Per la gestione delle query per le varie gestione è stato usato solo una classe.

### ***Database***

|  |  |
| --- | --- |
| **Nome** | Database |
| **Descrizione** | Questa classe rappresenta il gestore del pool di connessioni con il database. |
| **Signature dei metodi** | createDBConnection()  getConnection()  releaseConnection(Connection connection) |
| **Precondizione** |  |
| **Post-condizione** |  |
| **Invariante** |  |

### ***DatabaseQuery***

|  |  |
| --- | --- |
| **Nome** | DatabaseQuery |
| **Descrizione** | Questa classe rappresenta il gestore del database per ogni gestione. |
| **Nome delle query** | queryAddUtente  queryEliminaUtente  queryGetUtente  queryAddProdotto  queryEliminaProdotto  queryGetProdotti  queryGetMieiOrdini  queryCercaProdotto  queryGetProdottoById  queryGetProdottoByUser  queryAddCarrello  queryAddOrdine  queryGetCarrello  queryEliminaCarrello  queryGetNumeroProdotto  queryGetUtenti  queryGetAdmin |
| **Precondizione** |  |
| **Post-condizione** | **context:** gestioneUtente :: addUser(Utente utente) **post:** utente != null |
| **Invariante** |  |