Università degli Studi di Salerno

Corso di Ingegneria del Software

Project: PC Zone
Object Design Document
Versione 3.2

Data: 06/01/2023

Partecipanti

Nome	Matricola
Simone Scermino	0512110611
Roberto Andrei Miron	0512110581

Revision History

Data	Versione	Descrizione	Autore
01/12/2022	1.0	Inizio stesura ODD, indice	Simone Scermino, Roberto Andrei Miron
03/12/2022	1.1	Design goals	Simone Scermino, Roberto Andrei Miron
06/12/2022	2.0	Guidelines, References	Simone Scermino, Roberto Andrei Miron
09/12/2022	2.1	Packages, inizio class interfaces	Simone Scermino, Roberto Andrei Miron
15/12/2022	3.0	Fine contratti class interfaces	Simone Scermino, Roberto Andrei Miron
23/12/2022	3.1	Modifica contratti class interfaces	Simone Scermino, Roberto Andrei Miron
06/01/2023	3.2	Ulteriori modifiche class interfaces, modifica di alcune precondizioni	Simone Scermino, Roberto Andrei Miron

1. Introduction

PCZone si propone di rendere l'assemblaggio di un PC accessibile anche agli utenti meno esperti, offrendo loro la possibilità di entrare a far parte di una comunità appassionata nel campo. Inoltre, un altro obiettivo importante della piattaforma è di ridurre i costi dell'assemblaggio.

In questo documento, verranno dettagliatamente descritti i trade-off, le linee guida per la documentazione, implementazione, nomenclatura, interfacce delle classi e operazioni supportate, tipi di dati, parametri delle procedure e la suddivisione dei sottosistemi. Tutte queste informazioni saranno fondamentali per garantire una progettazione e implementazione efficace e coerente del sistema.

1.1 Object design trade-offs

Trade-off	Descrizione
Performance vs Scalabilità	Migliorare la concorrenza del sistema, e quindi il numero di utenti simultanei che possono accedervi influenzerebbe sulla performance del sito quindi il sistema dovrà dare precedenza alla Performance.
Complessità vs Semplicità	Un design complesso può fornire più funzionalità e flessibilità ma rende il sistema più difficile da capire, testare e mantenere. Il sistema dovrà dare la precedenza alla semplicità e quindi essere intuitivo e facile da usare per l'utenza meno esperta.
Costo vs Qualità	Utilizzare tecnologie costose garantisce una migliore qualità del sistema ma aumenta i costi del sistema e della manutenzione. Un sistema che usa tecnologie meno costose può ridurre i costi ma può anche compromettere il

	sistema. Nell'ambito della piattaforma, si è deciso di dare la precedenza al risparmio dei costi.
End User Criteria vs Funzionalità	Un design accattivante e un interfaccia intuitiva migliora l'esperienza per l'utente ma aumenta anche il tempo di sviluppo e il costo. Un design meno accattivante invece, può permettere di aumentare le funzionalità della piattaforma. Il sistema in questione darà la precedenza alle funzionalità nel caso in cui non si riuscirà ad implementare tutto entro i limiti di tempo.
Usabilità vs Manutenzione	Un design altamente user-friendly può richiedere più tempo per essere sviluppato e può essere più difficile da essere mantenuto. Il sistema cercherà di bilanciare entrambe ma darà la precedenza all'usabilità.

1.2 Interface documentation guidelines

Nomenclatura

I nomi dovrebbero essere descrittivi, di lunghezza medio-corta, non abbreviati e dovrebbero usare solo caratteri consentiti.

Variabili

I nomi delle variabili devono rispettare la notazione a cammello, iniziando quindi con la lettera minuscola per poi avere eventuali parole successive con la lettera maiuscola (es. provaVariabile).
Si preferisce effettuare la dichiarazione di esse ad inizio blocco, evitando di unire più definizioni nella stessa riga.
In caso di variabili statiche o costanti si può adottare l'uso di "_".

Metodi

Per permettere una identificazione migliorata dei metodi creati, si preferisce usare un verbo nel nome di essi, rispettando la notazione a cammello.

Come da convenzione si utilizza “get” o “set” rispettivamente per metodi che permettono l’accesso o la modifica alle variabili dell’oggetto interessato.
In caso di variabili locali al metodo, si preferisce istanziare esse appena prima del loro utilizzo, limitando la loro funzionalità a quell’unico scopo.

Classi e pagine

Per poter identificare al meglio le classi all’interno del codice, i loro nomi dovranno iniziare con la lettera maiuscola (come anche le parole successive nel nome). Devono poter rappresentare lo scopo della classe che rappresentano; ogni file .java deve contenere una sola classe.

1.3 Definitions, acronyms, and abbreviations

Acronimi:

- RAD: Requirements Analysis Document
- SDD: System Design Document
- ODD: Object Design Document

Abbreviazioni:

- DB: DataBase
- DBMS: DataBase Management System
- DAO: Data Access Object

1.4 References

- B. Bruegge, A. H. Dutoit, Object Oriented Software Engineering - Using UML, Pattern and Java, Prentice Hall, 3rd edition, 2009
- Documento SDD di progetto
- Documento RAD di progetto

1.5 Design Pattern

DAO

Nell’implementazione di PcZone è stato usato il design pattern chiamato DAO.

Il DAO è un livello intermedio tra l'applicazione e la fonte dei dati, che fornisce un'interfaccia per l'accesso ai dati e gestisce tutte le complessità coinvolte nell'esecuzione delle operazioni di accesso ai dati. Questo pattern è utile per la gestione della persistenza dei dati, poiché permette di isolare la logica di business dalle operazioni di accesso ai dati, rendendo il codice più facile da mantenere e modulare.
Inoltre, i DAO possono essere utilizzati per testare la logica di business senza accedere alla fonte dei dati reale, rendendo i test più veloci e meno complessi.
Nel caso del nostro sistema, il pattern sarà usato per gestire meglio la persistenza dei dati e quindi far comunicare le servlet con l'interfaccia dei vari DAO delle classi specifiche.

2. Packages

Nella suddivisione realizzata per le classi implementate si è deciso di rispettare l'architettura three-tier, evitando un'ulteriore suddivisione in base alle funzionalità del sistema.
Questo è stato fatto principalmente per poter avere una visione più chiara sulle funzionalità che ogni classe ha, senza creare troppi package e rischiare di rendere troppo intricata la navigazione del progetto.
Di seguito una breve descrizione dei package principali.

Data package

Qui sono presenti tutte le classi che riguardano il salvataggio di dati permanenti tramite il DBMS adottato. I DAO sono uno strumento ottimo per questo lavoro, come già precedentemente accennato, infatti qui ritroviamo tutti quelli adottati, insieme ai Bean, classi che fanno semplicemente da contenitore per le informazioni da dover gestire (ad esempio le informazioni del profilo utente).

Application package

Qui sono presenti tutte le classi che contengono la logica principale del sistema (infatti questo tier nell'architettura viene solitamente anche chiamato "logic tier"). Nel nostro caso, ovviamente, la logica è rappresentata dalle servlet.

Presentation package

Nella cartella WEB-INF sono presenti tutte le classi inerenti a questo package. Si tratta principalmente di pagine JSP o HTML e, come il nome del package suggerisce, contengono l'interfaccia grafica mostrata all'utente del sistema, mandando eventuali richieste al tier sottostante (application) per poter ricevere/spedire nuovi dati.

3. Class interfaces

Componente

Classe	
Nome	**Descrizione**
ComponenteModelDM	Classe che attraverso l'implementazione dell'interfaccia ComponenteModel gestisce i dati delle componenti presenti nel sistema
Metodi	
Context:ComponenteModelDM public syncronized in doSave(ComponenteBean product)	
Descrizione	Grazie a questo metodo, vengono salvati i dati delle componenti nel database
Pre-Condition	pre: product != null
Post-Condition	-
Context: ComponenteModelDM public synchronized ComponenteBean doRetrieveByKey(int IDComponente)	
Descrizione	Grazie a questo metodo, viene ritrovato una componente che ha come chiave l'ID richiesto
Pre-Condition	pre: IDComponente != null and IDComponente > -1
Post-Condition	-

Context: ComponenteModelDM public synchronized boolean doDelete(int IDComponente)	
Descrizione	Grazie a questo metodo, viene cancellata una componente che ha l'ID richiesto
Pre-Condition	pre: IDComponente != null and IDComponente > -1
Post-Condition	true if result == 1
Context: ComponenteModelDM public synchronized int doUpdate(int cancellato, int IDComponente)	
Descrizione	Grazie a questo metodo, viene aggiornata una componente che ha l'ID richiesto
Pre-Condition	pre: cancellato != null and IDComponente != null and IDComponente > -1
Post-Condition	-
Context: ComponenteModelDM public synchronized int doUpdate(ComponenteBean product)	
Descrizione	Grazie a questo metodo, viene aggiornata una componente richiesta
Pre-Condition	pre: product != null
Post-Condition	-
Context: ComponenteModelDM public synchronized ArrayList<ComponenteBean> doRetrieveAll(String order)	
Descrizione	Grazie a questo metodo, vengono restituite tutte le componenti presenti nel sistema
Pre-Condition	-

Post-Condition	-
Context: ComponentModelDM public synchronized ArrayList<ComponenteBean> doRetrieveByCond(String gen)	
Descrizione	Grazie a questo metodo, vengono restituite le componenti con il genere indicato
Pre-Condition	gen != null
Post-Condition	-
public synchronized ArrayList<ComponenteBean> doRetrieveBySearchTerm(String searchTerm)	
Descrizione	Grazie a questo metodo, vengono restituite le componenti contenenti nel titolo o nella descrizione la parola indicata
Pre-Condition	searchTerm != null
Post-Condition	-

Configurazione

Classe	
Nome	**Descrizione**
ConfigurazioneModelDM	Classe che attraverso l'implementazione dell'interfaccia ConfigurazioneModel gestisce i dati delle configurazioni dell'utente
Metodi	
Context:ConfigurazioneModelDM public synchronized int doSave(ConfigurazioneBean ordine)	

Descrizione	Grazie a questo metodo, vengono salvati i dati delle configurazioni nel database
Pre-Condition	pre: ordine != null
Post-Condition	-
Context:ConfigurazioneModelDM public synchronized ConfigurazioneBean doRetrieveByKey(int IDOrdine)	
Descrizione	Grazie a questo metodo, vengono restituiti gli ordini che hanno la chiave richiesta
Pre-Condition	pre: IDOrdine != null and IDOrdine > -1
Post-Condition	-
Context:ConfigurazioneModelDM public synchronized boolean doDelete(int IDOrdine)	
Descrizione	Grazie a questo metodo, viene cancellata la configurazione con l'id richiesto
Pre-Condition	pre: IDOrdine != null and IDOrdine > -1
Post-Condition	true if result == 1
Context:ConfigurazioneModelDM public synchronized ArrayList<ConfigurazioneBean> doRetrieveAll(String order)	
Descrizione	Grazie a questo metodo, vengono restituiti tutte le configurazioni presenti nel database
Pre-Condition	-
Post-Condition	-

Context:ConfigurazioneModelDM public synchronized ArrayList<ConfigurazioneBean> doRetrieveByCond(String nickname)	
Descrizione	Grazie a questo metodo, vengono restituite le configurazioni di un certo utente
Pre-Condition	pre:nickname != null
Post-Condition	-

Genere

Classe	
Nome	**Descrizione**
GenereModelDM	Classe che attraverso l'implementazione dell'interfaccia GenereModel gestisce i dati riguardanti i tipi di componenti
Metodi	
Context:GenereModelDM public synchronized int doSave(GenereBean gen)	
Descrizione	Grazie a questo metodo, vengono salvati i dati dei tipi delle componenti nel database
Pre-Condition	pre: gen != null
Post-Condition	-
Context:GenereModelDM public synchronized ArrayList<GenereBean> doRetrieveAll()	
Descrizione	Grazie a questo metodo, vengono restituiti tutti i generi presenti nel database
Pre-Condition	-

Post-Condition	-
Context:GenereModelDM public synchronized boolean doDelete(String gen)	
Descrizione	Grazie a questo metodo, viene cancellato il tipo di componente richiesto
Pre-Condition	pre: gen != null
Post-Condition	true if result == 1

Guida

Classe	
Nome	**Descrizione**
GuidaModelDM	Classe che attraverso l'implementazione dell'interfaccia GuidaModel gestisce i dati riguardanti le guide presenti sul sito
Metodi	
Context:GuidaModelDM public synchronized int doSave(GuidaBean news)	
Descrizione	Grazie a questo metodo, vengono salvati i dati delle guide nel database
Pre-Condition	pre: news != null
Post-Condition	-
Context:GuidaModelDM public synchronized int doUpdate(GuidaBean guida)	
Descrizione	Grazie a questo metodo, vengono aggiornati i dati delle guide nel database

Pre-Condition	pre: guida != null
Post-Condition	-
Context:GuidaModelDM public synchronized int doSaveSuggerisce(GuidaBean news,int prodottoID)	
Descrizione	Grazie a questo metodo, vengono aggiunti nel database le componenti associate alla guida
Pre-Condition	pre: news != null and prodottoID != null and prodottoID > -1
Post-Condition	-
Context:GuidaModelDM public synchronized int doUpdateSuggerisce(int guidaID,int prodottoID, int compID)	
Descrizione	Grazie a questo metodo, vengono aggiornate nel database le componenti associate alla guida
Pre-Condition	pre: news != null and prodottoID != null and prodottoID > -1
Post-Condition	-
Context: GuidaModelDM public synchronized GuidaBean doRetrieveByKey(int IDGuida)	
Descrizione	Grazie a questo metodo, vengono restituite le guide che hanno la chiave indicata
Pre-Condition	pre: IDGuida != null and and IDGuida > -1
Post-Condition	-
Context: GuidaModelDM public synchronized GuidaBean doRetrieveByTitolo(String titolo)	

Descrizione	Grazie a questo metodo, vengono restituite le guide che hanno il titolo indicato
Pre-Condition	pre: titolo != null
Post-Condition	-
Context: GuidaModelDM public synchronized boolean doDelete(int IDGuida)	
Descrizione	Grazie a questo metodo, viene cancellata una guida
Pre-Condition	pre: IDGuida != null and IDGuida > -1
Post-Condition	true if result == 1
Context: GuidaModelDM public synchronized ArrayList<GuidaBean> doRetrieveAll(String order)	
Descrizione	Grazie a questo metodo, vengono restituite tutte le guide presenti nel database
Pre-Condition	-
Post-Condition	-
Context: GuidaModelDM public synchronized ArrayList<ComponenteBean> doRetrieveByCond(int IDGuida)	
Descrizione	Grazie a questo metodo, vengono restituite le compontenti associate ad una guida
Pre-Condition	pre: IDGuida != null and IDGuida > -1
Post-Condition	-

Context: GuidaModelDM public synchronized ArrayList<SuggerisciBean> doRetrieveSuggerisceByCond(int IDGuida)	
Descrizione	Grazie a questo metodo, vengono restituite le associazioni di una certa guida
Pre-Condition	pre: IDGuida != null and IDGuida > -1
Post-Condition	-

Incluso

Classe	
Nome	**Descrizione**
InclusoModelDM	Classe che attraverso l'implementazione dell'interfaccia InclusoModel gestisce i dati delle associazioni tra le configurazioni e le componenti presenti all'interno di se stessa
Metodi	
Context: InclusoModelDM public synchronized int doSave(InclusoBean product)	
Descrizione	Grazie a questo metodo, vengono salvati i dati delle associazioni tra configurazioni e componenti nel Database
Pre-Condition	pre: product != null
Post-Condition	-
Context: InclusoModelDM public synchronized InclusoBean doRetrieveByKey(int IDComponente, int IDOrdine)	
Descrizione	Grazie a questo metodo, viene restituito l'associazione tra una componente e una configurazione dati i loro id

Pre-Condition	pre: IDComponente != null and IDComponente > -1 IDOrdine != null and IDOrdine > -1
Post-Condition	-
Context: InclusoModelDM public synchronized InclusoBean doRetrieveByKey(int IDComponente, String nickname)	
Descrizione	Grazie a questo metodo, viene restituita la configurazione di un certo utente
Pre-Condition	pre: IDComponente != null and IDComponente > -1 nickname != null
Post-Condition	-
Context: InclusoModelDM public synchronized boolean doDelete(int IDComponente, int IDOrdine)	
Descrizione	Grazie a questo metodo, viene cancellata una componente da una configurazione
Pre-Condition	pre: IDComponente != null and IDComponente > -1 IDOrdine != null and IDOrdine > -1
Post-Condition	true if result == 1
Context: InclusoModelDM public synchronized ArrayList<InclusoBean> doRetrieveByCond(int IDOrdine)	
Descrizione	Grazie a questo metodo, vengono restituite le componenti di una configurazione dato la chiave della configurazione
Pre-Condition	pre: IDOrdine != null and IDOrdine > -1
Post-Condition	-
Context: InclusoModelDM public synchronized ArrayList<InclusoBean> doRetrieveByCond(String username)	

Descrizione	Grazie a questo metodo, vengono restituite le componenti di una configurazione dato l'username dell'utente
Pre-Condition	pre: username != null
Post-Condition	-
Context: InclusoModelDM public synchronized ArrayList<InclusoBean> doRetrieveAll(int IDOrdine)	
Descrizione	Grazie a questo metodo, vengono restituite tutte le componenti di una configurazione
Pre-Condition	pre: IDOrdine != null and IDOrdine > -1
Post-Condition	-

Review

Classe	
Nome	**Descrizione**
ReviewModelDM	Classe che attraverso l'implementazione dell'interfaccia ReviewModel gestisce i dati delle recensioni delle componenti
Metodi	
Context: ReviewModelDM public synchronized int doSave(ReviewBean review)	
Descrizione	Grazie a questo metodo, vengono salvati i dati delle recensioni
Pre-Condition	pre: review != null
Post-Condition	-

Context: ReviewModelDM public synchronized ReviewBean doRetrieveByKey(int IDComponente,String nickname)	
Descrizione	Grazie a questo metodo, viene restituita la recensione di un certo utente su una certa componente
Pre-Condition	pre: nickname != null and IDComponente != null and IDComponente > -1
Post-Condition	-
Context: ReviewModelDM public synchronized boolean doDelete(int IDComponente,String nickname)	
Descrizione	Grazie a questo metodo, viene cancellata una recensione
Pre-Condition	pre: nickname != null and IDComponente != null and IDComponente > -1
Post-Condition	true if result == 1
Context: ReviewModelDM public synchronized ArrayList<ReviewBean> doRetrieveByCond(int IDComponente)	
Descrizione	Grazie a questo metodo, vengono restituite tutte le recensioni di una certa componente
Pre-Condition	pre: IDComponente != null and IDComponente > -1
Post-Condition	-

Wishlist

Classe

Nome	Descrizione
WishlistModelDM	Classe che attraverso l'implementazione dell'interfaccia WishlistModel gestisce i dati delle liste di favoriti
Metodi	
Context: WishlistModelDM public synchronized int doSave(WishlistBean wishlist)	
Descrizione	Grazie a questo metodo, vengono salvati i dati delle liste dei favoriti
Pre-Condition	pre: wishlist != null
Post-Condition	-
Context: WishlistModelDM public synchronized int doSave(int IDWishlist, String nickname, int IDComponente)	
Descrizione	Grazie a questo metodo, vengono salvati i dati delle liste dei favoriti, dati gli id della wishlist, il nickname dell'utente e l'id della componente
Pre-Condition	pre: nickname != null and IDWishlist != null and IDWishlist > -1 and IDComponente != null and IDComponente > -1
Post-Condition	-
Context: WishlistModelDM public synchronized WishlistBean doRetrieveByKey(String username)	
Descrizione	Grazie a questo metodo, viene restituita una wishlist di un utente
Pre-Condition	pre: username != null
Post-Condition	-
Context: WishlistModelDM public synchronized WishlistBean doRetrieveByKey(int IDWishlist)	

Descrizione	Grazie a questo metodo, viene restituita una lista di favoriti dato il suo id
Pre-Condition	pre: IDWishlist != null and IDWishlist > -1
Post-Condition	-
Context: WishlistModelDM public synchronized ContieneBean doRetrieveByKey(int IDWishlist, int IDComponente)	
Descrizione	Grazie a questo metodo, viene restituita una associazione tra una lista di favoriti e l'id della componente contenuta in essa
Pre-Condition	pre: IDWishlist != null and IDWishlist > -1 and IDComponente != null and IDComponente > -1
Post-Condition	-
Context: WishlistModelDM public synchronized boolean doDelete(int IDWishlist)	
Descrizione	Grazie a questo metodo, viene cancellata una lista di favoriti dato il suo id
Pre-Condition	pre: IDWishlist != null and IDWishlist > -1 and
Post-Condition	true if result == 1
Context: WishlistModelDM public synchronized boolean doDelete(String username)	
Descrizione	Grazie a questo metodo, viene cancellata una lista di favoriti dato l'username dell'utente
Pre-Condition	pre: username != null
Post-Condition	-

Context: WishlistModelDM public synchronized boolean doDelete(int IDWishlist, int IDComponente)	
Descrizione	Grazie a questo metodo, viene cancellata una lista di favoriti dato il suo id e l'id della componente che è contenuta in essa
Pre-Condition	pre: IDWishlist != null and IDWishlist > -1 and IDComponente != null and IDComponente > -1
Post-Condition	-
Context: WishlistModelDM public synchronized ArrayList<ContieneBean> doRetrieveAll(int IDWishlist)	
Descrizione	Grazie a questo metodo, vengono restituite tutte le componenti di una lista di favoriti
Pre-Condition	pre: IDWishlist != null and IDWishlist > -1
Post-Condition	-

Utente

Classe	
Nome	**Descrizione**
UserModelDM	Classe che attraverso l'implementazione dell'interfaccia UserModel gestisce i dati degli utenti e dei gestori
Metodi	
Context: UserModelDM public synchronized int doSave(UserBean user)	
Descrizione	Grazie a questo metodo, vengono salvati i dati degli utenti
Pre-Condition	pre: user != null

Post-Condition	-
Context: UserModelDM public synchronized int doUpdate(UserBean user)	
Descrizione	Grazie a questo metodo, vengono aggiornati i dati degli utenti
Pre-Condition	pre: user != null
Post-Condition	-
Context: UserModelDM public synchronized int doUpdatePass(UserBean user)	
Descrizione	Grazie a questo metodo, viene aggiornata la password di un utente
Pre-Condition	pre: user != null
Post-Condition	-
Context: UserModelDM public synchronized UserBean doRetrieveByKey(String nickname)	
Descrizione	Grazie a questo metodo, vengono restituiti i dati di un utente dato il suo nickname
Pre-Condition	pre: nickname != null
Post-Condition	-
Context: UserModelDM public synchronized UserBean doRetrieveByCond(String email,String password)	
Descrizione	Grazie a questo metodo, conferma l'esistenza dell'utente con l'email e password richiesta

Pre-Condition	pre: email != null and password != null
Post-Condition	-
Context: UserModelDM public synchronized boolean doDelete(String nickname)	
Descrizione	Grazie a questo metodo,viene cancellato un utente dal database
Pre-Condition	pre: nickname != null
Post-Condition	true if result == 1
Context: UserModelDM public synchronized Collection<UserBean> doRetrieveAll(String order)	
Descrizione	Grazie a questo metodo,vengono restituiti tutti gli utenti presenti nel database
Pre-Condition	-
Post-Condition	-

AggiungiRecensione

Classe	
Nome	**Descrizione**
AggiungiRecensioni	Classe che viene usata per l'aggiunta delle recensioni nelle pagine delle componenti
Metodi	
Context: AggiungiRecensioniServ protected void doPost(HttpServletRequest request, HttpServletResponse response)	

Descrizione	Questo metodo permette l'aggiunta di una recensione alla pagina della componente attraverso il passaggio dei parametri nella richiesta
Pre-Condition	pre: request != null and response != null and request.getParameter("titolo") != null and request.getParameter("testo") != null and request.getParameter("username") != null and
Post-Condition	-

CambioGenere

Classe	
Nome	**Descrizione**
CambioGenere	Classe che viene usata per il filtraggio delle componenti in base al tipo
Metodi	
Context:CambioGenereServ protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette il filtraggio dei componenti in base al tipo scelto
Pre-Condition	pre: request != null and response != null and request.getParameter("genere") != null
Post-Condition	-

CambioInformazioni

Classe	
Nome	**Descrizione**
CambioInformazioni	Classe che viene usata per cambiare le informazioni di base

	dell’utente, tranne la password
Metodi	
Context:CambioInformazioni protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette l’aggiornamento delle informazioni dell’utente
Pre-Condition	pre: request != null and response != null and request.getParameter(“email”) != null and request.getParameter(“descrizione”) != null and request.getParameter(“foto”) != null and
Post-Condition	-

CambioPassword

Classe	
Nome	**Descrizione**
CambioPassword	Classe che viene usata per cambiare la password dell’utente.
Metodi	
Context:CambioPassword protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette l’aggiornamento della password dell’utente
Pre-Condition	pre: request != null and response != null and request.getParameter(“psw2”) != null and request.getSesssion().getAttribute(“user”) != null
Post-Condition	-

EffettuaOrdine

Classe

Nome	Descrizione
EffettuaOrdine	Classe che viene per registrare una configurazione nel database.
Metodi	
Context:EffettuaOrdine protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette l'inserimento della configurazione nel database.
Pre-Condition	pre: request != null and response != null and request.getParameter("CPU") != null and request.getParameter("GPU") != null and request.getParameter("Storage") != null and request.getParameter("RAM") != null and request.getParameter("MOBO") != null and request.getParameter("Case") != null and request.getParameter("PSU") != null and request.getSesssion().getAttribute("carello") != null and request.getSesssion().getAttribute("user") != null
Post-Condition	-

Login

Classe	
Nome	**Descrizione**
Login	Classe che viene usata l'autenticazione dell'utente e del gestore
Metodi	
Context:Login protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette l'autenticazione dell'utente e del gestore

Pre-Condition	pre: request != null and response != null and request.getParameter(“email”) != null and request.getParameter(“password”) != null and request.getSesssion().getAttribute(“user”) != null
Post-Condition	-

Logout

Classe	
Nome	**Descrizione**
Logout	Classe che viene per usata per la disconnessione dell’’utente dall’account
Metodi	
Context:Logout protected void doGet(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette la disconnessione dell’’utente dall’account
Pre-Condition	pre: request != null and response != null and request.getSesssion() != null
Post-Condition	-

RegProdotto

Classe	
Nome	**Descrizione**
RegProdotto	Classe che viene usata per l’inserimento di una componente nel catalogo
Metodi	
Context:RegProdotto protected void doPost(HttpServletRequest request, HttpServletResponse response)	

Descrizione	Questo metodo permette l'inserimento di una componente nel database
Pre-Condition	pre: request != null and response != null and request.getParameter("nome") != null and request.getParameter("descrizione") != null andrequest.getParameter("prezzo") != null andrequest.getParameter("immagine") != null andrequest.getParameter("tipo") != null andrequest.getParameter("data") != null and request.getParameter("acquisto") != null
Post-Condition	-

ModifInformComp

Classe	
Nome	**Descrizione**
ModifInformComp	Classe che viene usata per la modifica delle informazioni di una componente
Metodi	
Context:ModifInformComp protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette l'aggiornamento delle informazioni di una componente
Pre-Condition	pre: request != null and response != null and request.getParameter("nome") != null and request.getParameter("descrizione") != null andrequest.getParameter("prezzo") != null andrequest.getParameter("immagine") != null andrequest.getParameter("tipo") != null andrequest.getParameter("data") != null and request.getParameter("acquisto") != null and request.getParameter("id") != null
Post-Condition	-

RegGuida

Classe	
Nome	**Descrizione**
RegGuida	Classe che viene usata per l'inserimento delle guide
Metodi	
Context:RegGuida protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette l'inserimento di guida
Pre-Condition	pre: request != null and response != null and request.getParameter("nome") != null and request.getParameter("descrizione") != null and request.getParameter("cpu") != null and request.getParameter("gpu") != null and request.getParameter("psu") != null and request.getParameter("storage") != null and request.getParameter("case") != null and request.getParameter("mobo") != null and request.getParameter("ram") != null and request.getParameter("id") != null
Post-Condition	-

ModifInformGuida

Classe	
Nome	**Descrizione**
ModifInformGuida	Classe che viene usata per l'aggiornamento delle informazioni di una guida
Metodi	
Context:ModifInformGuida protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette l'aggiornamento di una guida nel database

Pre-Condition	pre: request != null and response != null and request.getParameter(“nome”) != null and request.getParameter(“descrizione”) != null and request.getParameter(“cpu”) != null and request.getParameter(“gpu”) != null and request.getParameter(“psu”) != null and request.getParameter(“storage”) != null and request.getParameter(“case”) != null and request.getParameter(“mobo”) != null and request.getParameter(“ram”) != null and request.getParameter(“id”) != null
Post-Condition	-

Registration

Classe	
Nome	**Descrizione**
Registration	Classe che viene usata per la registrazione degli utenti nel sistema
Metodi	
Context:Registration protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette l’inserimento dei dati di un nuovo utente nel database
Pre-Condition	pre: request != null and response != null and request.getParameter(“nome”) != null and request.getParameter(“email”) != null and request.getParameter(“psw”) != null and request.getParameter(“username”) != null
Post-Condition	-

RegistrationAdmin

Classe	
Nome	**Descrizione**
RegistrationAdmin	Classe che viene usata per la registrazione dei gestori nel sistema
Metodi	
Context:RegistrationAdmin void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette l'inserimento dei dati di un nuovo gestore nel database
Pre-Condition	pre: request != null and response != null and request.getParameter("nome") != null and request.getParameter("email") != null and request.getParameter("psw") != null and request.getParameter("username") != null
Post-Condition	-

SceltaLoginControl

Classe	
Nome	**Descrizione**
SceltaLoginControl	Classe che viene usata per la scelta del ruolo del gestore
Metodi	
Context:SceltaLoginControl protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette la scelta del ruolo del gestore
Pre-Condition	pre: request != null and response != null and request.getParameter("scelta") != null and
Post-Condition	-

SearchBar

Classe	
Nome	**Descrizione**
SearchBar	Classe che viene usata per la barra di ricerca.
Metodi	
Context:SearchBar protected void doPost(HttpServletRequest request, HttpServletResponse response)	
Descrizione	Questo metodo permette di cercare varie componenti
Pre-Condition	pre: request != null and response != null and request.getParameter(“searchTerm”) != null and
Post-Condition	-