Università degli Studi di Salerno
Corso di Ingegneria del Software

Nome Progetto: Gametop
Titolo Documento: Object Design Document

Sommario

OBJECT DESIGN DOCUMENT

Manager:

Professori
Prof. De Lucia Andrea

Partecipanti:

Nome	Matricola
Vincenzo Tarantino	0512108025
Luigi Sodano	0512107590

Revision History:

Data	Versione	Descrizione	Autore
18/10/2023	1.0	Prima stesura del document problem statement	Team members
30/10/2023	2.0	Stesura del document RAD	Team members
15/11/2023	3.0	Revisione del document RAD	Team members
03/01/2024	3.1	Ultima revisione del document	Team members
20/11/2023	3.2	Stesura document System Design	Team members

23/01/2024	3.3	Revisione document System Design	Team members
03/12/2023	4	Stesura Object design document	Team members
23/01/2024	4.1	Ultima revisione del Object design document	Team members

1. Object Design Trade-offs

● Comprensibilità vs Tempo:

Il codice verrà scritto nella maniera più semplice possibile affiancato da commenti che ne facilitino la comprensione, questo per agevolare la fase di testing ed eventuali modifiche al codice.

● Interfaccia vs Usabilità:

L'interfaccia del sistema sarà intuitiva in modo da essere di facile utilizzo per l'utente finale, quest'ultimo sarà guidato, nella navigazione del sito e nell'utilizzo delle sue funzionalità, da form e pulsanti.

● Sicurezza vs Efficienza:

La sicurezza è uno dei requisiti non funzionali del sistema ed il sistema implementerà questo aspetto importante tramite un sistema di autenticazione tramite e-mail e password

1.2 Linee guida per la documentazione dell'interfaccia

- **Naming conventions**

Gli sviluppatori seguiranno le seguenti linee guida per la definizione delle interfacce:

- **Classi Java e Servlet:**
  - I nomi dovranno iniziare con la lettera maiuscola. Se il nome contiene più parole, ognuna di esse dovrà iniziare con una lettera maiuscola

- I nomi dovranno corrispondere alle informazioni e/o funzionalità che offre quella classe o Servlet

La dichiarazione di classe deve essere caratterizzata da:

1. Dichiarazione della classe pubblica
2. Dichiarazioni di costanti
3. Dichiarazioni di variabili di classe
4. Dichiarazione di variabili d'istanza
5. Costruttore.
6. Commento e dichiarazione dei metodi

- **Metodi:**
  - I nomi dovranno iniziare con la lettera minuscola
  - Se il nome contiene più parole, ognuna di esse dovrà iniziare con la lettera maiuscola
  - I nomi dovranno corrispondere alle informazioni e/o funzionalità che offre quel metodo. Si utilizzerà un verbo più eventualmente aggettivi
  - I nomi dei metodi per ottenere e settare attributi seguiranno la regola di nominazione getNomeVariabile e setNomeVariabile.

- **Variabili:**
  - I nomi delle variabili dovranno iniziare con la lettera minuscola
  - Se il nome contiene più parole, ognuna di esse dovrà iniziare con la lettera maiuscola

2. Packages

Analizziamo ora la scelta di suddivisione in sottosistemi del sistema e l'organizzazione del codice in file.

2.1 Divisione in pacchetti

Il sistema è diviso in packages nel modo seguente:

1)Ogni package conterrà tutti i componenti utili definire una funzionalità specifica del nostro sistema. Ad esempio il pacchetto gestioneAccount conterrà LoginControl.java, i relativi bean UtenteBean e PersonaleBean, LoginModeDS.java si troveranno nello stesso pacchetto perché hanno funzionalità dedicate alla gestione degli account.

1)JSP relative all'account

Nome Classe	account.jsp
Descrizione	Pagina che mostra all'utente la pagina relativa al suo account

Nome Classe	cambioPassword.jsp
Descrizione	Pagina che mostra all'utente l'area per inserire i dati per cambiare la password

Nome Classe	successoRegistrazione.jsp
Descrizione	Pagina che mostra all'utente l'avvenuto successo della registrazione.

Nome Classe	successoCambioPassword.jsp
Descrizione	Pagina che mostra all'utente l'avvenuto successo del cambio password.

Nome Classe	adminDirettore.jsp
Descrizione	Pagina che mostra le operazioni dedicate al gestore ruoli.

Nome Classe	login.jsp
Descrizione	Pagina che mostra all'utente la pagina di login

Nome Classe	loginError.jsp
Descrizione	Pagina che mostra all'utente la pagina di login non avvenuto per un errore

Nome Classe	loginPersonale.jsp
Descrizione	Pagina che mostra al personale la pagina di login

Nome Classe	logout.jsp
Descrizione	Pagina che gestisce il logout di un utente

Nome Classe	registrazione.jsp
Descrizione	Pagina che mostra all'utente la pagina che consente l'inserimento dei dati di registrazione

2)JSP relative agli Acquisti

Nome Classe	spedizione.jsp
Descrizione	Pagina che mostra all'utente la pagina di inserimento dati di spedizione

Nome Classe	pagamento.jsp
Descrizione	Pagina che mostra all'utente la pagina di inserimento dati di pagamento

Nome Classe	adminOrdini.jsp
Descrizione	Pagina che mostra le operazioni dedicate al gestore ordini.

Nome Classe	successoOrdine.jsp
Descrizione	Pagina che mostra l'avvenuto successo di un ordine.

3)JSP relative al carrello

Nome Classe	carrelloFine.jsp
Descrizione	Pagina che mostra all'utente il contenuto del proprio carrello

4)JSP relative ai prodotti

Nome Classe	prodotto.jsp
Descrizione	Pagina che mostra all'utente le specifiche di un prodotto

Nome Classe	console.jsp
Descrizione	Pagina che mostra all'utente i prodotti riguardanti la categoria console

Nome Classe	accessori.jsp
Descrizione	Pagina che mostra all'utente i prodotti riguardanti la categoria accessori

Nome Classe	videogiochi.jsp
Descrizione	Pagina che mostra all'utente i prodotti riguardanti la categoria videogiochi

Nome Classe	adminProdotti.jsp
Descrizione	Pagina che mostra le operazioni dedicate al gestore dei prodotti.

4)JSP generali

Nome Classe	index.jsp

Descrizione	Pagina che mostra all'utente la homepage del sito

Nome Classe	header.jsp
Descrizione	Pagina che mostra all'utente l'header ad inizio pagina

Nome Classe	headerNavBar.jsp
Descrizione	Pagina che mostra all'utente la barra di navigazione

Nome Classe	footer.jsp
Descrizione	Pagina che mostra all'utente il footer in fondo alla pagine

1) Package GestioneAccount

Nome Classe	UtenteBean.java
Descrizione	Questa classe rappresenta le informazioni di un utente

Nome Classe	PersonaleBean.java
Descrizione	Questa classe rappresenta le informazioni di un utente di tipo "personale"

Nome Classe	AccountControl.java
Descrizione	Questa classe è una servlet che si occupa di gestire le richieste web mostrando lo storico degli ordini effettuati da un utente. Fa uso della OrdiniModelDS.java per prendere gli ordini associati al cliente.

Nome Classe	CambiaPasswordControl.java
Descrizione	Questa classe è una servlet che si occupa di gestire le richieste web per modificare la password dell'utente. Fa uso della RegistrazioniModelDS.java per cambiare la password.

Nome Classe	GestorePersonaleControl.java
Descrizione	Questa classe è una servlet che si occupa di gestire le richieste web per la gestione del personale. Fa uso della GestorePersonaleModelDS.java per aggiungere, rimuovere o modificare i dati un personale.

Nome Classe	GestorePersonaleModelDS.java
Descrizione	Questa classe contiene i metodi che permettono di effettuare le operazioni di validazione, aggiunta, rimozione e modifica personale.

Nome Classe	LoginModelDS.java
Descrizione	Questa classe contiene i metodi che permettono di effettuare le operazioni di validazione e selezione dei dati dell'utente

Nome Classe	LoginControl.java

Descrizione	Questa classe è una servlet che si occupa di ricevere i dati di login, elaborarli e decidere se consentire o meno l'accesso all'area personale utilizzando i servizi di LoginModelDS.java e Sessione.

Nome Classe	LoginPersonaleControl.java
Descrizione	Questa classe è una servlet si occupa di ricevere i dati di login, elaborarli e decidere se consentire o meno l'accesso all'area personale utilizzando i servizi di GestorePersonaleModelDS.java e Sessione.

Nome Classe	LogoutControl.java
Descrizione	Questa classe è una servlet che si occupa di ricevere le richieste per invalidare la sessione per consentire il logout.

Nome Classe	RegistrazioneModelDS.java
Descrizione	Questa classe contiene i metodi che permettono di effettuare le operazioni di visualizzazione, aggiunta, cancellazione e modifica degli account utente.

Nome Classe	RegistrazioneControl.java
Descrizione	Questa classe è una servlet che si occupa di ricevere i dati di registrazione, elaborarli e decidere se consentire o meno la registrazione dell'utente. Fa uso dei servizi di RegistrazioneModelDS.java e di UtentiBean.java.

2) Package Gestione Carrello

Nome Classe	CarrelloControl.java
Descrizione	Questa classe è una servlet che si occupa di gestire le funzionalità che offre un carrello come la possibilità di aggiungere un prodotto al carrello, eliminarlo, svuotare il carrello e procedere all'ordine. Utilizza i GestioneCarrelloModelDS, ShopModelDS, OrdineModelDS e Carts

Nome Classe	Carts.java
Descrizione	Questa classe rappresenta il carrello come lista di oggetti. Fornisce i metodi per aggiungere prodotti al carrello, eliminarli, ottenere la lista di tutti i prodotti e svuitare la lista. Viene utilizzato da CarrellControl.java

Nome Classe	GestioneCarrelloModelDS
Descrizione	Questa classe contiene i metodi che permettono di effettuare le operazioni per rendere persistenti i singoli prodotti di un carrello. Contiene metodi per aggiungere ed eliminare singoli prodotti dalle tabelle Carrello e Storico oltre che a contenere i metodi per trovare i prodotti che fanno parte di un carrello con una determinata email e acquistati in base all'email.

Nome Classe	SessionCarrelloBean
Descrizione	Questa classe rappresenta il singolo prodotto del carrello in sessione. Viene usato da GestioneCarrelloModelDS per ricavare le informazioni per le operazioni sulle tabelle.

Nome Classe	CarrelloFineControl.java

Descrizione	Questa classe è una servlet che si occupa di gestire le funzionalità che offre un carrello come la possibilità di aggiungere un prodotto al carrello, eliminarlo, svuotare il carrello e procedere all'ordine. È simile a CarrelloControl ma rappresenta un carrello che gestisce la creazione di un ordine al momento del pagamento. Utilizza i GestioneCarrelloModelDS, ShopModelDS, OrdineModelDS e Carts

3) Gestione Prodotti

Nome Classe	ShopBean.java
Descrizione	Questa classe rappresenta le informazioni di un prodotto

Nome Classe	ShopModelDS.java
Descrizione	Questa classe è un DAO che contiene i metodi che permettono di effettuare le operazioni di visualizzazione di prodotti, quelli che permettono di aggiungere, modificare ed eliminare un prodotto dal catalogo ed anche quelli per cercare i prodotti in base alla categoria(Videogiochi, Console, Accessori).

Nome Classe	VideogiochiControl.java
Descrizione	Questa classe è una servlet che si occupa di gestire le richieste web per visualizzare dei prodotti della categoria Videogiochi. Fa uso di ShopModelDS.

Nome Classe	ConsoleControl.java
Descrizione	Questa classe è una servlet che si occupa di gestire le richieste web per visualizzare dei prodotti della categoria Console. Fa uso di ShopModelDS.

Nome Classe	AccessoriControl.java
Descrizione	Questa classe è una servlet che si occupa di gestire le richieste web per visualizzare dei prodotti della categoria Accessori. Fa uso di ShopModelDS.

Nome Classe	GestoreProdottiControl.java
Descrizione	Questa classe è una sevrlet che si occupa di gestire le richieste web per la gestione dei prodotti da parte di un utente "Personale" con ruolo gestore prodotti. Fa uso di ShopModelDS

4) Package Gestione Acquisti

Nome Classe	OrdineBean.java
Descrizione	Questa classe rappresenta le informazioni di un oggetto ordine.

Nome Classe	OrdineModelDS.java
Descrizione	Questa classe contiene il metodo per salvare un ordine nel database quando viene pagato.

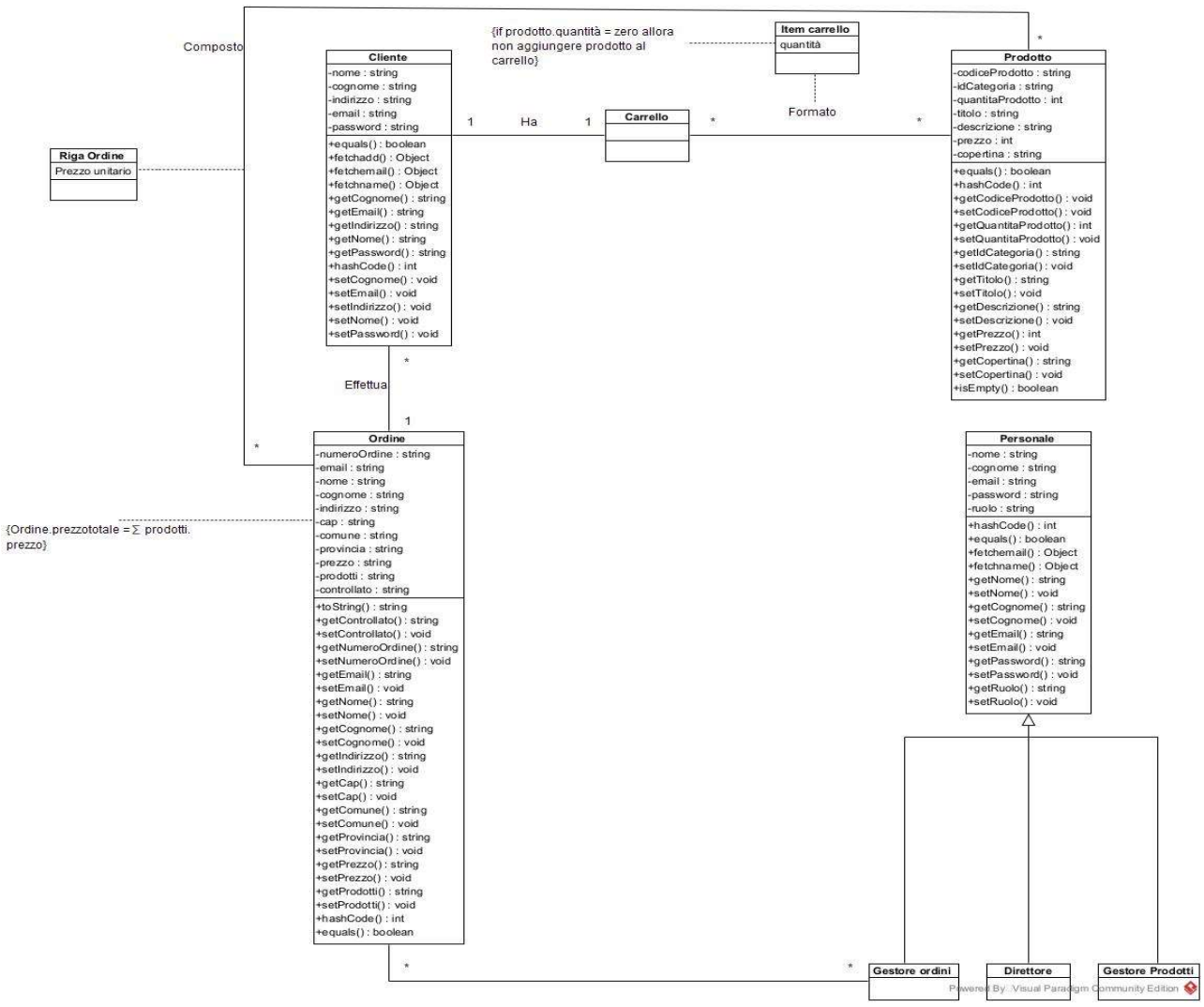
Nome Classe	GestoreOrdiniControl.java

Descrizione	Questa classe è una servlet che si occupa gestire le richieste web ricevendo i dati degli ordini, elaborarli e decidere se confermare, eliminare e modificare i dati dell'ordini. Fa uso dei servizi di GestioneOrdiniModelDS.java per effettuare le operazioni dette in precedenza.

Nome Classe	GestioneOrdiniModelDS.java
Descrizione	Questa classe è un DAO che permette di effettuare le operazioni riservate ad un utente "Personale" con ruolo gestore ordini.

Nome Classe	Spedizione Control
Descrizione	Questa classe è una servlet che si occupa di gestire le richieste web ricevendo i dati di spedizione e mettendoli nella sessione.

3. Diagramma delle classi



4. Interfaccia delle classi

Nome Classe:	**RegistrazioneModelDS**

Context:	RegistrazioneModelDS:doSave(item:UtenteBean)
Descrizione	Questo metodo consente di aggiungere un nuovo Utente all'interno della tabella Clienti del database.
Pre-condizione:	L'utente deve essere diverso da null e l'e-mail deve essere diversa da tutte le altre email appartenenti agli utenti già presenti nella tabella Clienti.
Post-condizione:	L'utente è visibile nella tabella Cliente e la sua e-mail è univoca.

Context:	RegistrazioneModelDS:doDelete(item:UtenteBean):Boolean
Descrizione	Questo metodo consente di rimuovere un utente all'interno della tabella Clienti del database restituendo true se è andato a buon fine, false altrimenti.
Pre-condizione:	L'utente deve essere diverso da null e la sua email deve essere presente all'interno di una tupla nella tabella Clienti.
Post-condizione:	Return true se la rimozione all'interno della tabella Clienti è andata a buon fine ed il numero delle tuple all'interno della tabella Prodotto sarà ridotto di uno, false altrimenti.

Context:	RegistrazioneModelDS:doUpdate(item:UtenteBean):Boolean
Descrizione	Questo metodo consente di aggiornare i dati dell'utente all'interno della tabella Clienti del database restituendo true se è andato a buon fine, false altrimenti.
Pre-condizione:	L'utente deve essere diverso da null e l'email del prodotto deve essere presente all'interno di una tupla nella tabella Cliente.
Post-condizione:	Return true se la modifica all'interno della tabella Cliente è andata a buon fine e aggiorna i campi del dell'utente passato come parametro, false altrimenti.

Context:	RegistrazioneModelDS:doUpdatePassword(item:UtenteBean):Boolean
Descrizione	Questo metodo consente di modificare la password dell'utente all'interno della tabella Clienti del database restituendo true se è andato a buon fine, false altrimenti.
Pre-condizione:	L'utente deve essere diverso da null e l'email del prodotto deve essere presente all'interno di una tupla nella tabella Cliente.
Post-condizione:	Return true se la modifica all'interno della tabella Cliente è andata a buon fine e aggiorna i campi del dell'utente passato come parametro, false altrimenti.

Context:	RegistrazioneModelDS:doRetrieveAll(order:String):Collection<UtenteBean>
Descrizione	Questo metodo consente di ottenere tutti gli utenti all'interno della tabella Clienti del database.
Pre-condizione:	
Post-condizione:	Return dell'insieme di tutte le tuple all'interno della tabella Clienti.

Context:	RegistrazioneModelDS:doRetrieveByKey(email:String):UtenteBean
Descrizione	Questo metodo consente di ottenere l'utente all'interno della tabella Clienti del database con l'e-mail specificata.
Pre-condizione:	e-mail deve essere diverso da null e deve essere presente all'interno di una tupla nella tabella Clienti.
Post-condizione:	Return dell'utente con email specificato dall'interno della tabella Clienti

Nome Classe:	**GestioneOrdiniModelDS**

Context:	GestioneOrdiniModelDS:ritornaTuttiGliOrdiniDaControllare(): Collection<OrdineBean>
Descrizione	Questo metodo consente di restituire tutte le tuple all'interno della tabella ordine del database con il parametro controllato=false.
Pre-condizione:	
Post-condizione:	Restituisce tutte le tuple della tabella ordine all'interno del dataBase.

Context:	GestioneOrdiniModelDS:doRetrieveByKey(code:String): OrdineBean
Descrizione	Questo metodo consente di ottenere l'ordine all'interno della tabella ordine del database con il codice specificato in code.
Pre-condizione:	Code diverso da null e code deve essere presente all'interno di una tupla nella tabella ordine.
Post-condizione:	Return dell'ordine con id specificato in code.

Context:	GestioneOrdiniModelDS:doDelete(item:OrdineBean):Boolean
Descrizione	Questo metodo consente di rimuovere un ordine all'interno della tabella Ordine del database restituendo true se l'eliminazione è andata a buon fine, false altrimenti.
Pre-condizione:	Item deve essere diverso da null e il numeroOrdine di item deve essere presente all'interno di una tupla nella tabella Ordine
Post-condizione:	Return true se la rimozione all'interno della tabella Ordine è andata a buon fine ed il numero delle tuple all'interno della tabella Prodotto sarà ridotto di uno, false altrimenti.

Context:	GestioneOrdiniModelDS:doUpdate(item:OrdineBean):Boolean
Descrizione	Questo metodo consente di aggiornare un ordine all'interno della tabella ordine restituendo true se l'aggiornamento è andato a buon fine, false altrimenti.
Pre-condizione:	Item deve essere diverso da null e il numeroOrdine di item deve essere presente all'interno di una tupla nella tabella ordine
Post-condizione:	Return true se la modifica all'interno della tabella ordine è andata a buon fine e aggiorna i campi dell'ordine passato come parametro, false altrimenti.

Context:	GestioneOrdiniModelDS:confermaOrdine(item:OrdineBean) :Boolean
Descrizione	Questo metodo consente di aggiornare un ordine all'interno della tabella ordine settando il parametro controllo a true. Restituisce true se è andato a buon fine, false altrimenti.
Pre-condizione:	Item deve essere diverso da null e il numeroOrdine dell'ordine deve essere presente all'interno di una tupla nella tabella ordine
Post-condizione:	Viene aggiornato il campo controllato a true nella tupla della tabella ordine e ritorna true.

Context:	GestioneOrdiniModelDS:ritornaTuttiOrdiniUtente(email:String) :Collection<OrdineBean>
Descrizione	Questo metodo consente di ottenere tutti gli ordini della tabella Ordine fatti dall'utente che ha l'email uguale a quella passata come parametro.
Pre-condizione:	email deve essere diverso da null e email deve essere appartenente ad un utente della tabella Clienti
Post-condizione:	Ottengo una collezione di OrdineBean che rappresenta tutti gli ordini fatti dall'utente che ha come email quella passata

Nome Classe:	**OrdineModelDS**

Context:	OrdineModelDS:SalvaOrdine(ordini: OrdineBean)
Descrizione	Questo metodo consente di aggiungere un nuovo ordine all'interno della tabella ordine del database.
Pre-condizione:	L'ordine deve essere diverso da null e il numerordine deve essere diverso da tutti gli altri numerordine appartenenti agli ordini già presenti nella tabella ordine.
Post-condizione:	L'ordine è visibile nella tabella ordine e il numerordine è univoco

Nome Classe:	**ShopModelDS**

Context:	ShopModelSD:doSave(item:ShopBean)
Descrizione	Questo metodo consente di aggiungere un nuovo prodotto all'interno della tabella Prodotto del database.
Pre-condizione:	Il prodotto deve essere diverso da null e il codiceProdotto deve essere diverso da tutti gli altri codiceProdotto appartenenti ai prodotti già presenti nella tabella vestito.
Post-condizione:	Il prodotto è visibile nella tabella Prodotto e il codiceProdotto è univoco

Context:	ShopModelSD:doRetrieveAll(order:String):Collection<ShopBean>
Descrizione	Questo metodo consente di ottenere l'insieme dei prodotti all'interno della tabella Prodotto del database in base ad un certo ordine.
Pre-condizione:	order diverso da null e deve essere uguale ad "ASC", "DESC" o ad una stringa vuota
Post-condizione:	Return dell'insieme di tutte le tuple all'interno della tabella Prodotto

Context:	ShopModelSD:doRetrieveAllConsole(order:String): Collection< ShopBean >
Descrizione	Questo metodo consente di ottenere l'insieme dei prodotti console all'interno della tabella Prodotto del database.
Pre-condizione:	
Post-condizione:	Return dell'insieme di tutte le tuple con categoria console all'interno della tabella Prodotto

Context:	ShopModelSD:doRetrieveAllAccessori(order:String): Collection<ShopBean>
Descrizione	Questo metodo consente di ottenere l'insieme dei prodotti accessori all'interno della tabella Prodotto del database.
Pre-condizione:	
Post-condizione:	Return dell'insieme di tutte le tuple con categoria accessori all'interno della tabella Prodotto

Context:	ShopModelSD:doRetrieveAllVideogiochi(order:String): Collection< ShopBean >
Descrizione	Questo metodo consente di ottenere l'insieme dei prodotti videogiochi all'interno della tabella Prodotto del database.
Pre-condizione:	
Post-condizione:	Return dell'insieme di tutte le tuple con categoria videogiochi all'interno della tabella Prodotto

Context:	ShopModelSD:doRetrieveByKey(code:String): ShopBean
Descrizione	Questo metodo consente di ottenere il prodotto all'interno della tabella Prodotto del database con codiceProdotto specificato.
Pre-condizione:	code diverso da null e code deve essere presente all'interno di una tupla nella tabella Prodotto
Post-condizione:	Return del prodotto con id specificato dall'interno della tabella Prodotto

Context:	ShopModelSD:doDelete(item:ShopBean):Boolean
Descrizione	Questo metodo consente di rimuovere il prodotto passato della tabella Prodotto del database restituendo true se l'eliminazione è andato a buon fine, false altrimenti.
Pre-condizione:	item deve essere diverso da null e il codiceProdotto del prodotto deve essere presente all'interno di una tupla nella tabella Prodotto
Post-condizione:	Return true se la rimozione all'interno della tabella Prodotto è andata a buon fine ed il numero delle tuple all'interno della tabella Prodotto sarà ridotto di uno, false altrimenti.

Context:	ShopModelSD:doUpdate(item: ShopBean):Boolean

Descrizione	Questo metodo consente di aggiornare il prodotto all'interno della tabella Prodotto del database restituendo true se l'aggiornamento è andato a buon fine, false altrimenti.
Pre-condizione:	prodotto deve essere diverso da null e il codiceVestito del prodotto deve essere presente all'interno di una tupla nella tabella Prodotto
Post-condizione:	Return true se la modifica all'interno della tabella Prodotto è andata a buon fine e aggiorna i campi del prodotto passato come parametro, false altrimenti.

Nome Classe:	**GestorePersonaleModelDS**

Context:	GestorePersonaleModelDS:stampaTuttoIlPersonale():Collection<PersonaleBean>
Descrizione	Questo metodo consente di ottenere l'insieme del personale all'interno della tabella Personale del database.
Pre-condizione:	
Post-condizione:	Return dell'insieme di tutte le tuple all'interno della tabella Personale

Context:	GestorePersonaleModelDS:doRetrieveByKey(code:String):PersonaleBean
Descrizione	Questo metodo consente di ottenere il personale all'interno della tabella Personale del database con email specificata.
Pre-condizione:	code diverso da null e code deve essere presente all'interno di una tupla nella tabella Personale
Post-condizione:	Return del personale con email specificata all'interno della tabella Personale

Context:	GestorePersonaleModelDS:inserisciPersonale(item:PersonaleBean)
Descrizione	Questo metodo consente di aggiungere un nuovo personale all'interno della tabella Personale del database.
Pre-condizione:	item deve essere diverso da null e l'email deve essere diversa da tutte le altre email appartenenti ai personali già presenti nella tabella Personale.
Post-condizione:	Il personale è visibile nella tabella Personale e l'email è univoca.

Context:	GestorePersonaleModelDS:doUpdate(item:PersonaleBean):Boolean

Descrizione	Questo metodo consente di aggiornare il personale all'interno della tabella Personale del database restituendo true se è andato a buon fine, false altrimenti.
Pre-condizione:	item deve essere diverso da null e l'email dell'item deve essere presente all'interno di una tupla nella tabella Personale
Post-condizione:	Return true se la modifica all'interno della tabella Personale è andata a buon fine e aggiorna i campi del personale passato come parametro.

Context:	GestorePersonaleModelDS:doDelete(item:PersonaleBean):Boolean
Descrizione	Questo metodo consente di rimuovere il personale all'interno della tabella Personale del database restituendo true se è andato a buon fine, false altrimenti.
Pre-condizione:	item deve essere diverso da null e l'email del personale deve essere presente all'interno di una tupla nella tabella Personale
Post-condizione:	Return true se la rimozione all'interno della tabella Personale è andata a buon fine ed il numero delle tuple all'interno della tabella Personale sarà ridotto di uno, false altrimenti.

Nome Classe:	**GestioneCarrelloModelDS**

Context:	GestioneCarrelloModelDS:doSave(carrello:SessionCarrello)
Descrizione	Questo metodo consente di aggiungere un nuovo elemento del carrello all'interno della tabella Carrello del database.
Pre-condizione:	Il SessionCarrello deve essere diverso da null.
Post-condizione:	Il SessionCarrello è visibile nella tabella Carrello

Context:	GestioneCarrelloModelDS:trovaCarrello(code:String):Carrello
Descrizione	Questo metodo consente di ottenere il carrello all'interno della tabella Carrello del database con email specificata.
Pre-condizione:	code diverso da null e code deve essere presente all'interno di una tupla nella tabella Carrello
Post-condizione:	Return del carrello con email specificata all'interno della tabella Carrello

Context:	GestioneCarrelloModelDS:doDelete(carrello:Carrello):Boolean
Descrizione	Questo metodo consente di rimuovere un prodotto dal carrello all'interno della tabella Carrello del database restituendo true se è andato a buon fine, false altrimenti.
Pre-condizione:	carrello deve essere diverso da null e l'email del carrello deve essere presente all'interno di una tupla nella tabella Carrello
Post-condizione:	Return true se la rimozione all'interno della tabella Carrello è andata a buon fine ed il numero delle tuple all'interno della tabella Carrello sarà ridotto di uno, false altrimenti.

Context:	GestioneCarrelloModelDS:deleteAll(carrello:Carrello):Boolean
Descrizione	Questo metodo consente di rimuovere il carrello all'interno della tabella Carrello del database restituendo true se è andato a buon fine, false altrimenti.
Pre-condizione:	carrello deve essere diverso da null e l'email del carrello deve essere presente all'interno di una tupla nella tabella Carrello
Post-condizione:	Return true se la rimozione all'interno della tabella Carrello è andata a buon fine ed il numero delle tuple all'interno della tabella Carrello sarà ridotto di uno, false altrimenti.

Nome Classe:	**LoginModelDS**

Context:	LoginModelDS: doRetrieveByKeyPersonale (code:String)
Descrizione	Questo metodo consente di ottenere il personale all'interno della tabella Personale del database con email specificata.
Pre-condizione:	Personale deve essere diverso da null e il personale con email cercata deve essere presente al'interno di una tupla nella tabella Personale
Post-condizione:	Return del personale con email specificata dall'interno della tabella Personale

Context:	LoginModelDS: doRetrieveByKey(code:String)
Descrizione	Questo metodo consente di ottenere il cliente all'interno della tabella Cliente del database con email specificata.
Pre-condizione:	cliente deve essere diverso da null e il cliente con email cercata deve essere presente al'interno di una tupla nella tabella cliente.
Post-condizione:	Return del Cliente con email specificato dall'interno della tabella Cliente