Laurea Triennale in Informatica - Università di Salerno
Corso di Ingegneria del Software - Prof.ssa F. Ferrucci

Object Design Document

GameHub

Riferimento	C01_ODD_V1.0
Versione	1.0
Data	10/01/2021
Destinatario	Top Management
Presentato da	Gerardo Brescia, Domenico D'Alessandro, Roberto Esposito, Francesco Mattina, Francesco Maria Rastelli, Andrea Terlizzi
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
05/12/2020	0.1	Inizio Stesura	Tutti
05/12/2020	0.2	Aggiunta Introduzione	FM, DD
05/12/2020	0.3	Aggiunta Componenti Off-The-Shelf	RE, GB
05/12/2020	0.4	Aggiunta Design Patterns	FMR, AT
10/12/2020	0.5	Aggiunta Packages	TUTTI
12/12/2020	0.6	Aggiunta Class Interfaces	TUTTI
15/12/2020	0.7	Aggiunta Class Diagram	TUTTI
03/01/2021	0.8	Revisone Class Diagram	TUTTI
05/01/2021	0.9	Revisione “Componenti off the shelf”.	FMR
10/01/2021	1.0	Revisione finale	TUTTI

Sommario

1. Introduzione

1.1 Object design trade-offs

Durante la fase di analisi e di progettazione sono stati individuati diversi compromessi per lo sviluppo del sistema. Inoltre, anche nella fase dell'Object Design sorgono diversi trade-offs di progettazione analizzati nel corso di questa sezione del documento:

Readability vs. Release Time

Il tempo di rilascio della piattaforma è uno degli obiettivi prioritari che sono stati individuati. A tal proposito non verrà commentata ogni singola riga del codice con descrizioni dettagliate ma, nonostante ciò, ci si propone di rendere il codice comprensibile e manutenibile. Lo si farà seguendo lo standard proposto da Google per la formattazione del codice, senza influenzare negativamente il tempo necessario per il rilascio del software.

Buy vs. Build

L'idea è quella di riutilizzare il più possibile soluzioni off-the-shelf, scegliendole in modo oculato al fine di trarne il maggior numero di vantaggi possibili. Ci si propone, quindi, di utilizzare soluzioni precostruite data la varietà di tools per lo sviluppo di applicazioni web. La scelta sarà effettuata basandosi sulle conoscenze dei membri del team evitando così che il tempo di apprendimento della nuova tecnologia non risulti svantaggioso rispetto alla sua effettiva utilità.

1.1.1 Componenti off-the-shelf

Per facilitare e velocizzare lo sviluppo della piattaforma GameHub, saranno utilizzate diverse componenti off-the-shelf di seguito riportate.

Sezione Applicazione Java EE

Bootstrap

Bootstrap è una raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web la quale contiene modelli di progettazione basati su HTML e CSS. All'interno del sistema Bootstrap verrà utilizzato per le componenti dell'interfaccia grafica.

È stato, inoltre, individuato il seguente tema: Link

Tale tema sarà integrato all'interno dell'applicazione effettuando le opportune modifiche.

In quanto la modifica di un tema già esistente risulta essere meno onerosa rispetto a realizzarne uno completamente da zero, ottenendo però gli stessi risultati.

jQuery & AJAX

Per permettere all'interfaccia grafica di rispondere alle azioni dell'utente in modo rapido ed efficiente e per migliorare la user experience durante l'interazione dell'utente con la piattaforma verranno utilizzati jQuery e AJAX.

jQuery è una libreria di JavaScript per applicazioni web il cui obiettivo è quello di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML, nonché semplificare le funzionalità di AJAX.

AJAX (Asynchronous JavaScript and XML) è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive, basandosi su uno scambio di informazioni in background fra client e server.

JSTL

JSTL è una libreria inclusa come componente della piattaforma di sviluppo per applicazioni web Java EE ed è un'estensione di JSP. Tale libreria verrà utilizzata per facilitare la composizione dinamica delle pagine che vengono mostrate all'utente.

Java Mail

JavaMail API è un framework che permette la creazione e l'invio di mail attraverso applicazioni scritte in Java ed è incluso sia all'interno di Java SE sia Java EE. Tale libreria verrà utilizzata per l'invio delle mail.

MailUtility

MailUtility è una classe che fa uso della JavaMail API e fornisce un metodo sendMail() che astrae le operazioni da effettuare per inviare una mail.

JUnit

JUnit è un framework di Unit testing per il linguaggio di programmazione Java. Tale libreria verrà utilizzata per effettuare i test di unità.

MySql-JDBC

MySQL Connector/J (JDBC) è il driver che permette l'interoperabilità fra Java e MySQL. Viene impiegato per permettere alla WebApp di interfacciarsi col Database, e dunque compiere operazioni CRUD.

ConPool

ConPool è una classe che permette di ottenere connessioni dal database attraverso un metodo getConnection(), astraendo JDBC.

Selenium

Selenium è un framework java che permette l'automatizzazione di operazioni su pagine web. Viene impiegato per effettuare i test delle pagine che compongo la View.

Mockito

Mockito è un framework di mocking che permette di effettuare richieste HTTP, creare risposte HTTP e sessioni fittizie, manipolandone parametri e contenuto, per poter testare componenti web java, nel nostro caso web servlets.

Spring

Spring è un framework java pensato per lo sviluppo di applicazioni EE. Da noi viene impiegato nel testing di integrazione, e nel testing di unità delle servlet.

Sezione Intelligenza Artificiale

Java Embedded Python (JEP)

Java Embedded Python è una libreria che permette di eseguire in modo "embedded" un interprete CPython all'interno della JVM. Questo garantisce la possibilità di comunicazione tra le classi Java del model e gli script Python che interrogano i modelli di machine learning, in modo totalmente trasparente.

OpenCSV

OpenCSV è una libreria che permette di eseguire in modo estremamente rapido ed efficiente operazioni su file CSV anche dalla struttura complessa, permettendo inoltre di esportare i dati presi da una query su un database, direttamente in formato CSV (programmaticamente). Viene utilizzata per salvare e leggere i dati degli utenti.

Scikit-learn

Scikit-learn è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python. Tale libreria verrà utilizzata per facilitare lo sviluppo di algoritmi di Machine Learning.

Pandas

Pandas è una libreria open source di Python che permette di effettuare operazioni CRUD sui dati. Viene impiegata per gestire i file con estensione '.csv'.

Pickle

Pickle è una libreria Python che permette di serializzare e deserializzare oggetti Python. Viene impiegata per serializzare il modello di Machine Learning allenato, in modo da non doverlo ri-allenare ogni volta che è richiesto il suo utilizzo.

Regex

Regex è una libreria Python per la gestione di espressioni regolari. Viene da noi utilizzata nell'ambito della formattazione dei dati da dare in input ad il modello di Machine Learning, per filtrare, sulla base delle nostre esigenze, alcuni dati in formato stringa.

1.1.2 Design Patterns

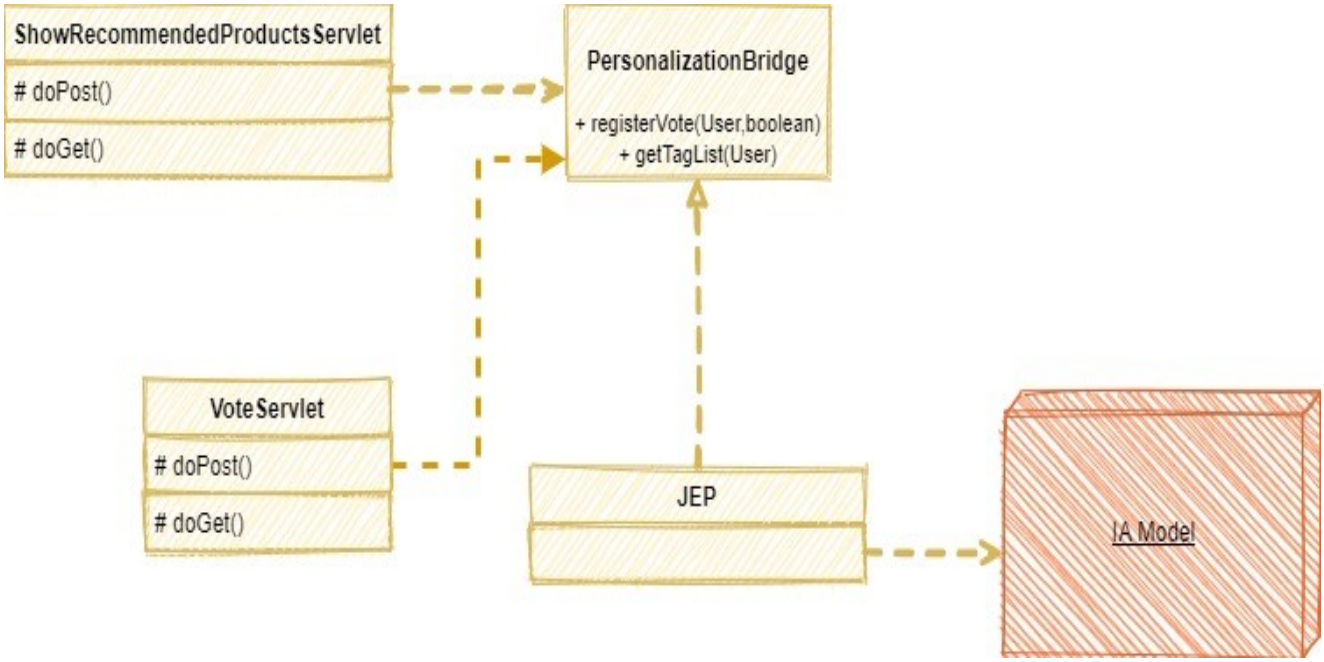
Per agevolare lo sviluppo è stato deciso di adottare dei Design Pattern rientranti fra quelli elencati dalla GoF e altri relativi al mondo delle applicazioni enterprise.

Adapter:

Il pattern in questione, individuato dalla GoF, viene impiegato quando è necessaria una nuova implementazione di un'interfaccia.

Sarà sfruttato allo scopo di creare un'interfaccia tra script Python (utilizzati per la realizzazione dei modelli di Machine Learning) e il resto dell'applicazione, scritta in Java.

Il wrapper è rappresentato dalle classi che prevedono l'utilizzo della libreria JEP, in particolare la classe PersonalizationBridge. Si noti che il nome PersonalizationBridge non fa riferimento in alcun modo al design pattern Bridge.

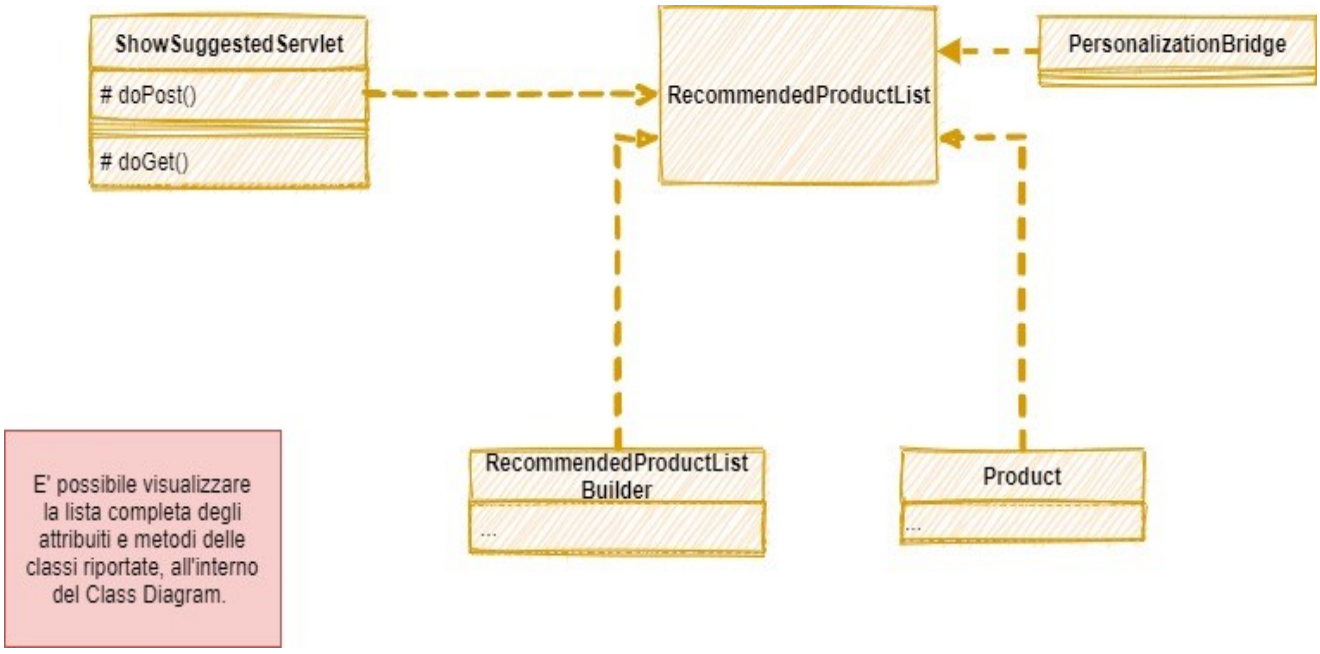


Proxy:

Tale pattern strutturale presentato dalla GoF viene utilizzato quando un oggetto non risulta immediatamente disponibile al consumo, in quanto l'operazione di creazione risulta onerosa.
Pertanto, viene utilizzato un oggetto proxy che condivide l'interfaccia con l'oggetto reale, ma non l'implementazione.
L'oggetto proxy viene "sostituito" con l'oggetto reale al momento della richiesta di una delle operazioni che non supporta.

Il DP in questione viene da noi impiegato per realizzare le funzionalità di personalizzazione. In particolare, un oggetto proxy viene impiegato come sostituto della lista dei prodotti consigliati dal sistema.

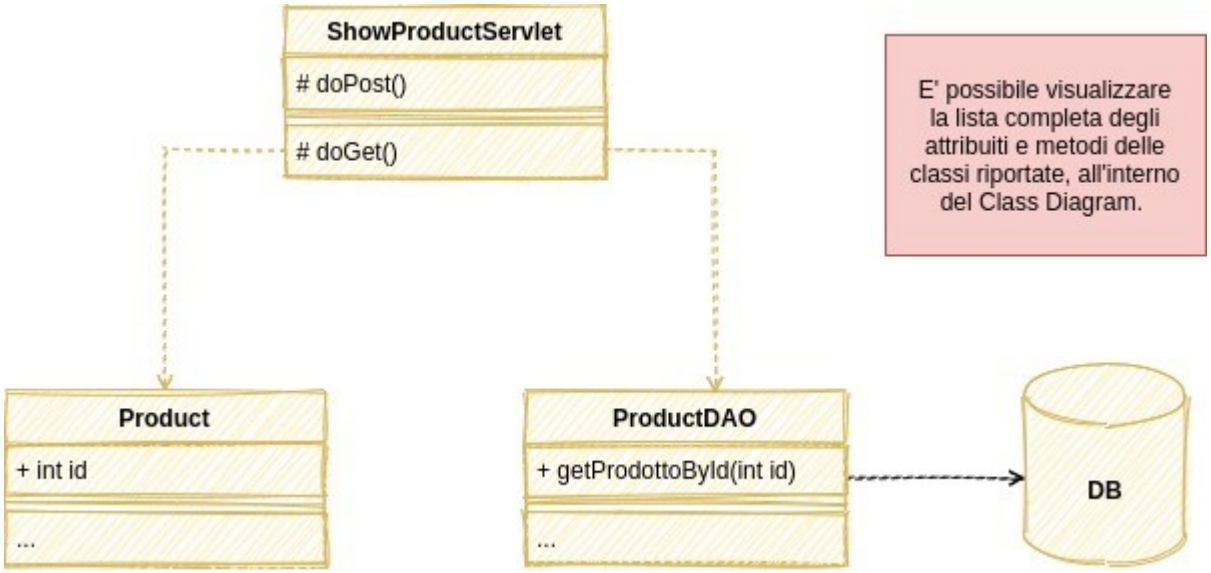
L'oggetto proxy viene impiegato in quanto la creazione della lista dei prodotti consigliati risulta un'operazione complessa, data la necessità di interrogare il modello di Machine Learning.



Data Mapper (DAO pattern):

Questo pattern, proposto da Martin Fowler, si pone come obiettivo quello di separare gli oggetti che rappresentano la logica di business dell'applicazione dall'informazione persistente.
Ciò permette di disaccoppiare la logica di data access dalla logica applicativa per garantire una maggiore manutenibilità, estensibilità e comprensibilità del codice.

Gli oggetti di business sono rappresentati da java beans mentre i corrispettivi DAO rappresentano la loro interfaccia per operazioni CRUD verso il database.



Singleton:

Il design pattern creazionale singleton, ideato dalla GoF, ha come scopo garantire che possa esistere una ed una sola istanza di una determinata classe allo stesso tempo, ad esempio per garantire che le operazioni che essa prevede possano essere eseguite da un solo utente alla volta. Prevede pertanto un costruttore privato, ed un metodo statico getInstance(), che ne restituisce l'istanza. Esistono molte implementazioni di questo design pattern, ma quella da noi scelta è quella di Bill Pugh, che prevede la presenza di una classe Helper interna alla classe del singleton, con un campo statico contenente un'istanza dello stesso. Il metodo getInstance() accede e restituisce tale istanza, ed in tal modo, l'istanza viene creata solo quando viene richiesta, e ne potrà esistere una ed una sola allo stesso tempo, trattandosi di un campo statico. Si è scelto di utilizzare questo design pattern nella classe PersonalizationBridge, che prevede inoltre metodi synchronized, allo scopo di evitare race condition sull'utilizzo dei file CSV, e garantire che un solo utente alla volta possa accedere alle operazioni.

Observer:

Pattern comportamentale identificato dalla GoF.

Questo DP viene da noi utilizzato al momento della realizzazione delle UI web, che saranno event-driven.

Gli 'observer' saranno rappresentati dai listener JavaScript, che resteranno in attesa della generazione di eventi da parte dei 'subject', rappresentati dagli elementi HTML con i quali l'utente interagirà.

1.2 Linee guida per la documentazione dell'interfaccia

Le seguenti linee guida saranno visionate degli sviluppatori al fine di comprendere la struttura implementativa inclusiva delle funzionalità di ogni componente:

1.2.1 Linee Guida Java

Le classi Java saranno scritte seguendo le convenzioni relative allo standard Google Java (Link alla documentazione), a questo standard sarà apportata la seguente modifica (approvata e condivisa dall'intero team):

- Le istruzioni o i blocchi racchiusi all'interno di un altro blocco saranno indentate di 4 unità (nell'esempio la dichiarazione della stringa è indentata di 4 unità rispetto alla firma del metodo main).

```
public static void main(final String[] args) {

    String c = "Prova";

}
```

Le restanti regole che verranno seguite saranno, come specificato in precedenza quelle corrispondenti allo standard Google. Di seguito sono specificate quelle di maggior rilevanza:

- La parentesi "{" aperta si trova alla fine della stessa linea della firma del blocco al quale si riferisce.
- Non devono esserci spazi a separare il nome del metodo e la "(" contente la lista dei parametri passati.

```
public static void main(final String[] args) {
```

- La parentesi "}" chiusa deve trovarsi allo stesso livello di indentazione del blocco al quale si riferisce. (Nell'esempio la parentesi graffa chiusa si trova allo stesso livello di public)

```
public static void main(final String[] args) {

    String c = "Prova";

}
```

- I nomi delle classi seguiranno la notazione Camel Case che consiste nello scrivere parole composte o frasi unendo tutte le parole tra loro, ma lasciando le loro iniziali maiuscole. (Esempio "ProvaClasse")

- I nomi dei metodi seguiranno ugualmente la notazione Camel Case ma la prima parola inizierà con una lettera minuscola. (Esempio "metodoProva")

```
public class ProvaClasse {

    public void metodoProva() {

        System.out.println("ProvaClasse");

    }

}
```

1.2.2 Linee Guida JavaScript

Lo standard sintattico di JavaScript seguirà quello descritto da Google.

1.2.3 Linee Guida JSP

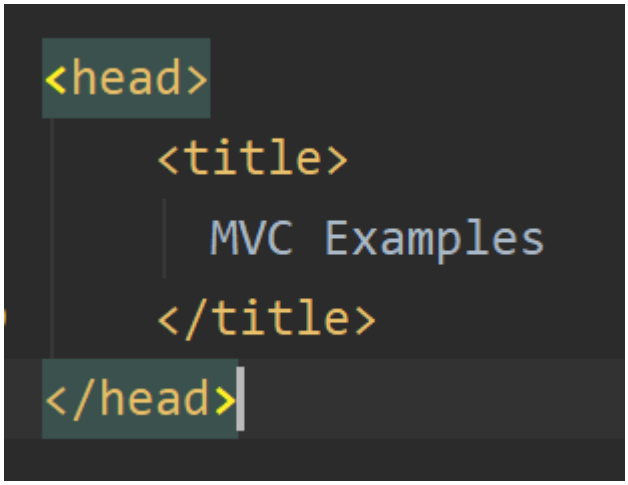
Ogni pagina JSP dovrà seguire le seguenti regole:

- Comprendere esclusivamente codice HTML relativo alla versione 5, codice JSTL, ed eventualmente codice CSS relativo alla versione 3 e JavaScript (Vedere le rispettive linee guida).

1.2.4 Linee Guida HTML

Anche per il codice HTML occorrerà utilizzare un'indentazione appropriata per facilitare la lettura del codice. Di seguito sono riportate le regole che verranno seguite:

- Un tag contenuto all'interno di un altro tag sarà indentato di 4 unità rispetto al tag più esterno.
- Ogni tag di chiusura si deve trovare allo stesso livello di quello di apertura corrispondente.
- Non verranno utilizzati tag considerati deprecati dalla W3School.



1.3.5 Linee Guida CSS

Ogni foglio di stile dovrà seguire le seguenti regole:

- È necessario attenersi a quello che è lo standard sintattico e semantico delle regole di Bootstrap.

1.3.6 Linee Guida SQL

Il nome delle tabelle dovranno devono essere scritte interamente in minuscolo.
Il nome degli attributi deve iniziare con la lettera minuscola. Se il nome della tabella risulta composto da più parole allora verrà seguito lo standard sintattico CamelCase (come per il nome degli attributi delle classi Java).

```
create table digitalbelonging
(
    digitalProduct int not null,
    category varchar(45) not null,
    primary key (digitalProduct, category),
    constraint DigitalBelonging_digitalproduct_id_fk
        foreign key (digitalProduct) references digitalproduc
            on update cascade on delete cascade,
    constraint DigitalBelonging_order_id_fk
        foreign key (category) references category (name)
            on update cascade on delete cascade
);
```

1.3 Definizioni, acronimi, e abbreviazioni

DP (Design Pattern): Si tratta di una descrizione o modello logico da applicare per la risoluzione di un problema che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo del software.

MVC: Model-view-controller è un pattern architetturale molto diffuso nello sviluppo di sistemi software.

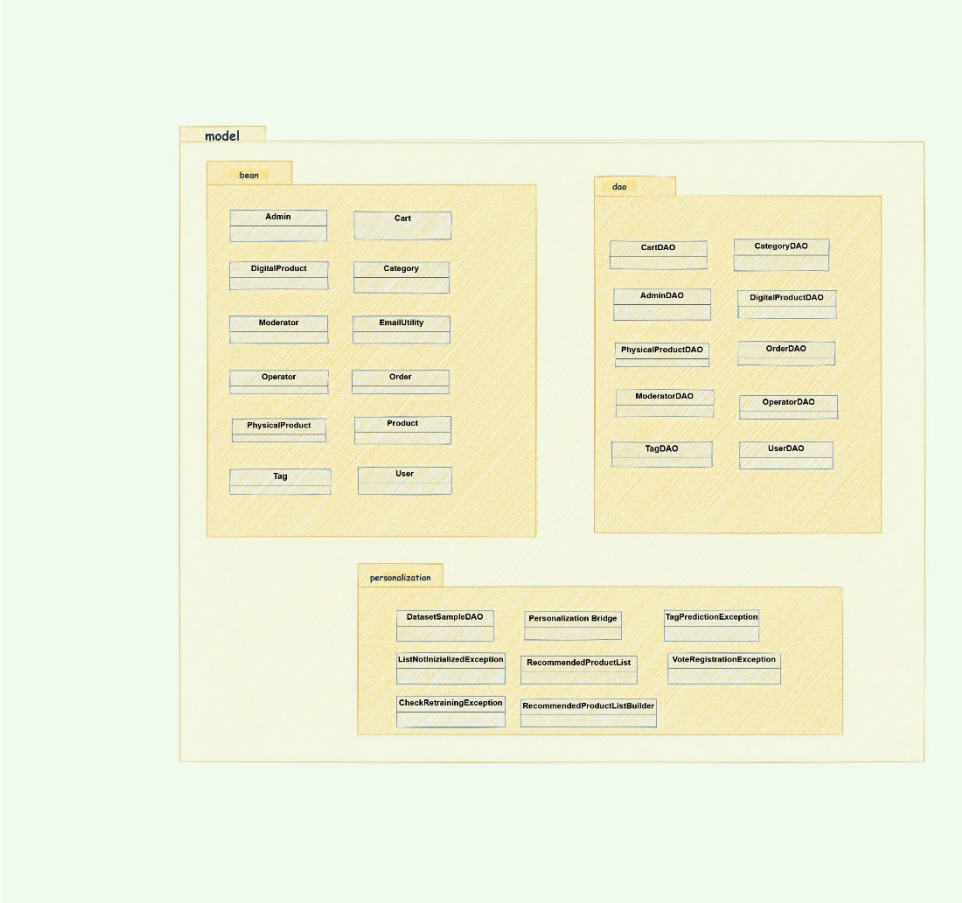
DAO (Data Access Object): è un pattern architetturale per la gestione della persistenza.

GoF: sigla usata in ingegneria del software per riferirsi agli autori del libro Design Patterns: Elements of Reusable Object-Oriented Software.

2. Packages

La divisione in package è molto legata alla divisione in sottosistemi effettuata nel documento di System Design, pertanto nella divisione in package si evidenzia l'utilizzo dell'architettura MVC. In questo capitolo andremo ad analizzare nello specifico i package, evidenziando le dipendenze che li collegano.

2.1 model



Il package model offre le classi per l'implementazione della profilazione e quindi quelle classi che permettono di consigliare all'utente dei prodotti. Questo package inoltre offre le classi DAO e i javaBeans. La suddivisione all'interno del package model seguirà questa separazione logica.

Il package model è suddiviso in ulteriori tre packages:

- bean
- dao
- personalization

bean

Il package bean contiene le classi javaBeans che offrono parte della logica di business ed inoltre rappresentano la mappatura object-oriented dell'informazione persistente all'interno del database.

Il package bean è composto da 12 javaBeans:

- Admin è il bean che rappresenta l'admin ed ha dipendenza interna al package con la classe Moderator, essendo sottoclasse di quest'ultima.
- Cart è il bean che rappresenta il carrello ed ha dipendenza interna al package con la classe Product, con una relazione molti a molti, e con User, attraverso una relazione uno ad uno.
- PhysicalProduct è il bean che rappresenta il prodotto fisico ed ha dipendenza interna al package con la classe Product in quanto sua sottoclasse.
- DigitalProduct è il bean che rappresenta il prodotto digitale ed ha dipendenza interna al package con la classe Product in quanto sua sottoclasse.
- Product è il bean che rappresenta il prodotto ed ha dipendenza interna al package con le classi Category e Tag, attraverso delle relazioni molti a molti.
- Tag è la classe che rappresenta un tag. Non ha dipendenze.
- Category è la classe che rappresenta una categoria. Non ha dipendenze.
- User è la classe che rappresenta un utente generico.
- Moderator e Operator rappresentano rispettivamente moderatore ed operatore e sono entrambe sottoclassi di User.
- Order è il bean che rappresenta un ordine effettuato da un utente. Pertanto, presenta una serie di prodotti al proprio interno, creando una relazione molti a molti, e tiene traccia dello User a cui si riferisce, con una relazione uno ad uno, e dell'eventuale operatore che ha approvato l'ordine, con una relazione uno ad 1.

dao

Il package dao offre le classi data mapper che permettono l'interazione con il database.

Il package dao è composto da 10 classi:

- CartDAO permette operazioni CRUD relative al carrello ed ha dipendenza con le classi User, DigitalProduct, PhysicalProduct, Cart contenute all'interno del package model. Ha dipendenza interna al package con le classi UserDAO, DigitalProductDAO, PhysicalProductDAO.
- CategoryDAO permette operazioni CRUD relative alle categorie ed ha dipendenza con la classe Category contenuta all'interno del package model.
- AdminDAO permette operazioni CRUD relative agli Admin ed ha dipendenza con la classe Admin contenuta all'interno del package model.bean, e con ModeratorDAO.
- DigitalProductDAO e PhysicalProductDAO permettono operazioni CRUD relative alle due tipologie di prodotti. Pertanto, hanno dipendenza con le classi DigitalProduct, PhysicalProduct, Product, Category e Tag, contenute all'interno del package model.bean.
- TagDAO permette operazioni CRUD relative ai tag ed ha dipendenza con la classe Tag contenuta all'interno del package model.bean.
- ModeratorDAO permette operazioni CRUD relative ai moderatori ed ha dipendenza con la classe Moderator, User e UserDAO, contenute all'interno del package model.bean e model.dao.
- OperatorDAO permette operazioni CRUD relative agli Operator; ha dipendenza con le classi Operator, User contenute all'interno del package model.bean, e con UserDAO.
- OrderDAO permette operazioni CRUD relative all'Order (bean) ed ha dipendenza con le classi User, DigitalProduct, PhysicalProduct, Cart, Order, Operator contenute all'interno del package model.bean. Ha dipendenza interna al package con le classi UserDAO, DigitalProducDAO, PhysicalProductDAO, OperatorDAO.
- UserDAO permette operazioni CRUD relative agli user ed ha dipendenza con la classe User contenuta all'interno del package model.bean

personalization

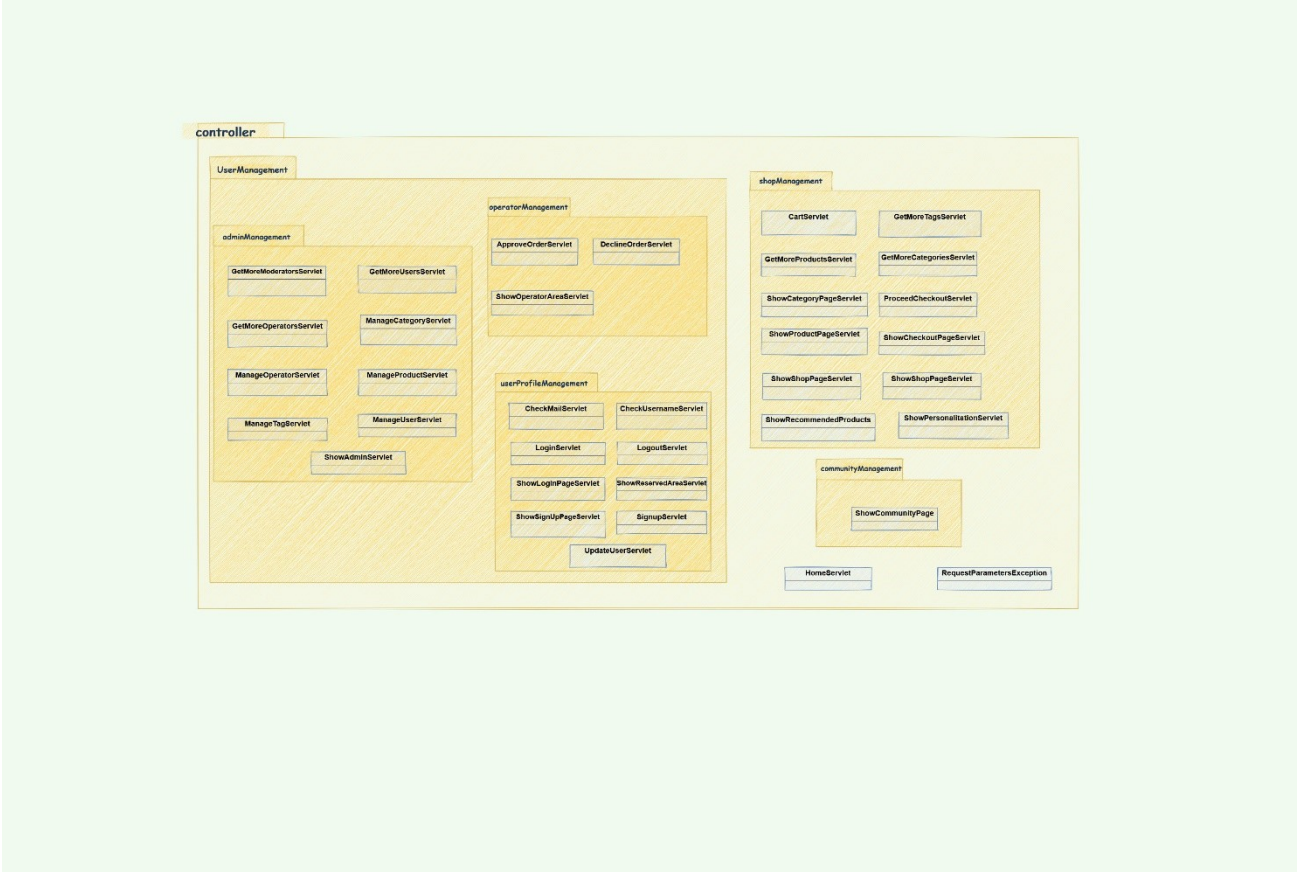
Il package personalization è composto dalle classi che si occupano di offrire i servizi di personalizzazione dell'esperienza utente. Pertanto, sono in esso presenti classi dedicati alla comunicazione con gli script Python (e quindi al modello di Machine Learning), classi che comunicano con i dataset CSV, e classi che modellano le componenti di personalizzazione (es. lista dei prodotti consigliati). Il package è composto da 7 classi:

- DatasetSampleDAO, che permette l'estrazione delle informazioni relative agli ordini di un utente dal database, per creare un nuovo campione da inserire nel dataset, o da sottoporre al modello di Machine Learning, salvandolo su un file buffer in formato CSV.
- PersonalizationBridge, singleton che utilizza la libreria JEP per fare da ponte tra gli script Python che comunicano ed interrogano il modello di Machine Learning, ed il resto dell'applicazione. Ha dipendenze con User, Tag e TagDAO.
- ReccomendedProductListBuilder, interroga il database, sulla base delle informazioni fornite da PersonalizationBridge e costruisce la lista dei prodotti consigliati effettiva, dipende da Tag, oltre che da User e Product, ed è composta da un'istanza di PhysicalProductDAO ed una di DigitalProductDAO.
- ReccomendedProductList, classe proxy che modella la lista dei prodotti consigliati. Alla sua creazione, essa è vuota, e solo quando viene richiesto esplicitamente l'accesso agli elementi della lista, richiama il PersonalizationBridge (come variabile statica) e ReccomendedProductListBuilder (con cui intrattiene una relazione uno ad uno), che interrogando i modelli di ML ed il database, creano la lista dei prodotti consigliati effettiva. Dipende da Tag, oltre che da User e Product.

2.2 controller

Il package controller è composto da classi che permettono di gestire le operazioni richieste dall'utente, richiama le classi del model per ottenere le informazioni richieste, ed infine inoltra queste informazioni alle JSP del package view, per poterle visualizzare.

(Link all'immagine)



Il package **controller** è suddiviso in ulteriori 3 packages:

- userManagement: si compone di tre package (adminManagement, operatorManagement, userProfileManagement).
- shopManagement: racchiude tutte le funzionalità relative allo shop.
- CommunityManagement.

Il package contiene anche una servlet:

- HomeServlet, servlet utile alla visualizzazione della homepage del sito internet, e che si occupa, quando instanziata, di inizializzare il ServletContext con la lista delle categorie, e dei prodotti "nuovi", contrassegnati con il tag "New". Ha dipendenze con Tag, TagDAO, Category, CategoryDAO, Product, PhysicalProduct, DigitalProduct, DigitalProductDAO e PhysicalProductDAO.

Ed un'eccezione:

- RequestParametersException, che viene lanciata da una servlet se le vengono passati dei parametri errati, od in presenza di parametri mancanti. Da essa dipendono tutte le servlet presenti nei sotto-package del package controller.

userManagement

Offre tutte le funzionalità relative agli utenti. All'interno di questo package le funzionalità sono ulteriormente divise in modo da riunire nello stesso pacchetto le classi che sono legate da una forte dipendenza da un punto di vista semantico.

Il package **adminManagemen**t è composto da 12 Servlet:

- GetMoreModeratorsServlet è la Servlet che permette di ottenere un certo numero di Moderator dal database, ed è utilizzata principalmente per chiamate AJAX; dipende dal package model dalle classi Moderator, ModeratorDAO.
- GetMoreAdminsServlet è la Servlet che permette di ottenere un certo numero di Admin dal database, ed è utilizzata principalmente per chiamate AJAX; dipende dal package model dalle classi Admin, AdminDAO.
- GetMoreUsersServlet è la Servlet che permette di ottenere un certo numero di User dal database, ed è utilizzata principalmente in chiamate AJAX; dipende dal package model, in particolare dalle classi User, UserDAO.
- GetMoreOperatorsServlet è la Servlet che permette di ottenere un certo numero di Operator da database, ed è utilizzata principalmente in chiamate AJAX; dipende dal package model, in particolare dalle classi Operator ed OperatorDAO.
- ManageCategoryServlet è la servlet che permette l'aggiunta, rimozione e modifica delle categorie è dipendente dal package model, in particolare le classi Category e CategoryDAO.
- ManageAdminsServlet è la servlet che permette di gestire l'aggiunta, e la rimozione degli admin, ed ha dipendenza con il package model con le classi Admin, Moderator, AdminDAO e ModeratorDAO.
- ManageModeratorServlet è la servlet che permette di gestire l'aggiunta e la rimozione dei moderatori, ed ha dipendenza con il package model con le classi Moderator, ModeratorDAO, User e UserDAO.
- ManageOperatorServlet è la servlet che permette di gestire l'aggiunta e la rimozione degli operatori, e dipende dal package model, in particolare le classi Operator, User, UserDAO ed OperatorDAO.
- ManageProductServlet è la servlet che permette di gestire l'aggiunta, rimozione, modifica dei prodotti, e dipende dal package mode, in particolare dalle classi Category, CategoryDAO, Tag, TagDAO, DigitalProduct, DigitalProductDAO, PhysicalProduct e PhysicalProductDAO.
- ManageTagServlet è la servlet che permette di gestire l'aggiunta, rimozione, modifica dei tag, ed ha dipendenza con il package model, in particolare con le classi Tag e TagDAO.

- ManageUserServlet è la servlet che permette di gestire la rimozione degli User, ed ha dipendenza con il package model, in particolare con le classi User e UserDAO.
- ShowAdminServlet è la Servlet che permette di visualizzare tutte le informazioni relative ai prodotti agli user, alle categorie, ai tags, agli operatori, ai moderatori. Pertanto, questa classe è collegata con il package model con le classi User, UserDAO, Category, CategoryDAO, Tag, TagDAO, Operator, OperatorDAO, Moderator, ModeratorDAO, Admin, AdminDAO. Questa servlet dipende anche dalla jsp AdminArea che si trova all'interno del package view.

Il package **operatorManagement** è composto da 3 Servlet:

- ApproveOrderServlet è la servlet che permette di approvare un ordine, riceve i comandi e restituisce le risposte attraverso la jsp OperatorArea contenuta all'interno del package view. Dipende inoltre dalle classi Operator, OperatorDAO, OrderDAO, Order, User, UserDAO.
- DeclineOrderServlet è la servlet che permette di declinare un ordine, riceve i comandi e restituisce le risposte attraverso la jsp OperatorArea contenuta all'interno del package view. Dipende inoltre dalle classi OrderDAO, Order.
- ShowOperatorAreaServlet è la servlet che permette di visualizzare l'area di un operatore, riceve i comandi e restituisce le risposte attraverso la jsp OperatorArea. Dipende inoltre dalle classi Order, User, OrderDAO.

Il package **userProfileManagement** è composto da 9 servlet:

- CheckMailServlet è la servlet che permette di controllare se una mail è valida e non esiste già un utente con la stessa mail. Dipende dalla UserDAO contenuta all'interno del package model.
- CheckUsernameServlet è la servlet che permette di controllare se un username è valida e non esiste già un utente con lo stesso username. Dipende dalla UserDAO contenuta all'interno del package model.
- LoginServlet è la servlet che permette di effettuare il login all'interno della piattaforma. Dipende dalle classi User, UserDAO, ModeratorDAO, OperatorDAO, AdminDAO all'interno del package model. Riceve e restituisce informazioni tramite la jsp Login presente nel package view, e fa il redirect alla homepage se il login va a buon fine.
- LogoutServlet è la servlet che permette di effettuare il logout all'interno della piattaforma. Fa il redirect alla homepage una volta effettuato il logout.
- ShowLoginPageServlet è la servlet che permette di mostrare la pagina di Login, pertanto è legata al package view con la jsp login.
- ShowReservedAreaServlet è la servlet che permette di mostrare la pagina dell'area riservata, pertanto è legata al package view con la jsp ReservedArea. Ha inoltre dipendenza con User, Order ed OrderDAO.
- ShowSignupPageServlet è la servlet che permette di mostrare la pagina per la registrazione, pertanto, è collegata al package view con la jsp Signup.

- SignupServlet è la servlet che permette la registrazione all'interno della piattaforma è collegata al package view in particolare con la jsp Signup. Dipende inoltre da User e UserDAO.
- UpdateUserServlet è la servlet che permette di aggiornare uno user è pertanto dipendente dalla classe UserDAO contenuta all'interno del package model.dao, oltre che da User, contenuta in model.bean.

shopManagement

Offre tutte le funzionalità relative allo shop, questo package è formato da 11 Servlet:

- CartServlet è la servlet che permette di gestire il carrello, dipende pertanto dalle classi Cart, CartDAO, DigitalProductDAO, PhysicalProductDAO, DigitalProduct, PhysicalProduct e User contenute nel package model. Inoltre, dipende dalla jsp cart contenuta nel package view.
- GetMoreCategoriesServlet è la servlet che permette di ottenere un certo numero di Category dal database; utilizzata principalmente per chiamate AJAX, ha dipendenza con il package model, in particolare con le classi Category e CategoryDAO.
- GetMoreProductsServlet è la servlet che permette di ottenere un certo quantitativo di Product dal database; utilizzata principalmente per chiamate AJAX, ha dipendenza con il package model, in particolare con le classi DigitalProductDAO, PhysicalProductDAO e Product.
- GetMoreTagsServlet è la servlet che permette di ottenere un certo quantitativo di Tag da database; utilizzata principalmente per chiamate AJAX, ha dipendenza con il package model, in particolare con le classi Tag e TagDAO.
- ProceedCheckoutServlet è la Servlet che permette l'acquisto di un carrello dipende dalla jsp purchaseConfirmed contenuta nel package view. Inoltre, ha dipendenze con il package model, in particolare con le classi UserDAO, CartDAO, OrderDAO, Cart, User, Order, Product, DigitalProduct, DigitalProductDAO, PhysicalProduct e PhysicalProductDAO.
- ShowCategoryPageServlet, ShowCheckoutPageServlet, ShowProductServlet, ShowShopPageServlet che permettono la visualizzazione rispettivamente di Category, Checkout, Products e la pagina. Dipendono da classi all'interno di model e view. In particolare, ShowCategoryPageServlet dipende da Category e CategoryDAO e la jsp category, ShowCheckoutPageServlet dipende da Cart e dalla jsp checkout, ShowProductsServlet dipende dalla jsp product, da Product, DigitalProduct, DigitalProductDAO, PhysicalProduct e PhysicalProductDAO.
- ShowRecommendedProducts permette di visualizzare la pagina dei prodotti consigliati. Dipende dalla jsp recommendedProducts.jsp e da RecommendedProductList.
- VotePersonalizationServlet permette di votare la personalizzazione, con un voto positivo o negativo. Dipende da RecommendedProductList.

communityManagement

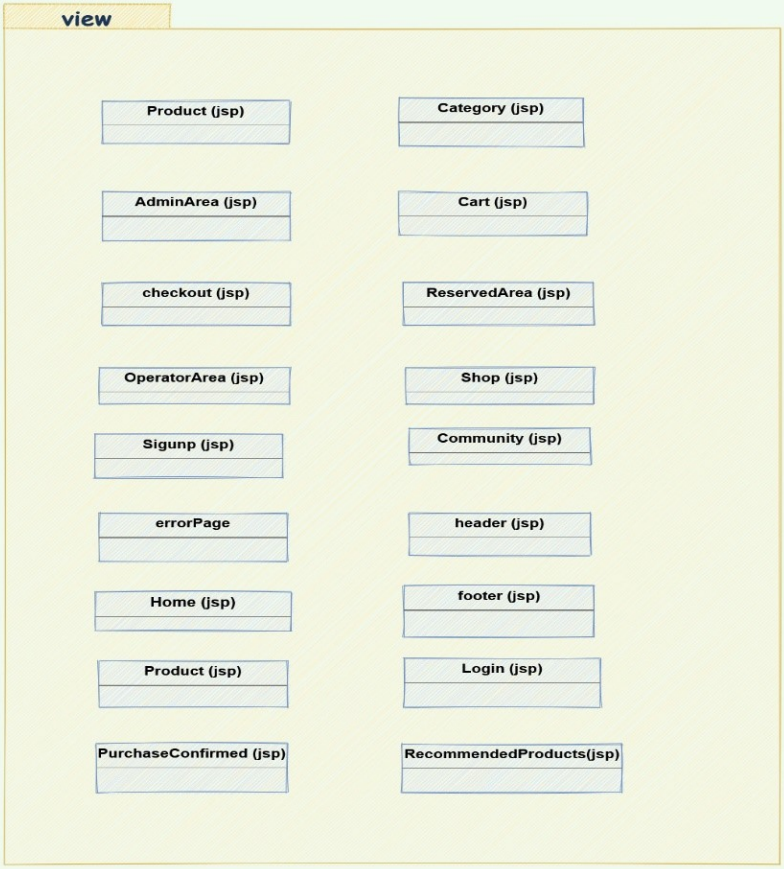
La community sarà rilasciata in una fase successiva, a questo proposito si è deciso di predisporre un package che in futuro racchiuderà tutte le servlet che gestiranno l'interazione dell'utente con le jsp relative all'area community.

Momentaneamente il package è composto da un'unica Servlet ShowCommunityPage che permette di mostrare la pagina che annuncia la futura release della sezione community e che dipende dalla jsp Community contenuta all'interno del package view.

2.3 view

Il package view è formato interamente da jsp che permetteranno all'utente di interagire con il sistema.

Si è scelto di non suddividere ulteriormente questo package in sotto-package per motivi relativi alla complessità d'implementazione, e del sistema in generale.



Tutte le jsp hanno dipendenza interna con le jsp header a footer.

La jsp dell'header ha inoltre a sua volta una dipendenza con l'errorPage

Veniamo ora alle dipendenze di ciascuna jsp, con il package model.bean:

- AdminArea (jsp): User, PhysicalProduct, DigitalProduct, Tag, Category, Admin, Moderator ed Operator;
- Cart (jsp): Cart, Product, DigitalProduct e PhysicalProduct;
- Category (jsp): Category;
- Checkout (jsp): Cart e Product;
- Home (jsp): Product;
- OperatorArea (jsp): Order e User;
- product (jsp): Product, PhysicalProduct, DigitalProduct, Tag e Category;
- recommendendProducts (jsp): Product;
- shop(jsp): Product;
- ReservedArea (jsp): Order e User;

2.4 Organizzazione dei file

Come lo standard java impone, il progetto è suddiviso in package, corrispondenti ciascuno ad una directory, ed ogni package conterrà un sottoinsieme delle classi, ognuna delle quali sarà contenuta in un file apposito. Il package delle jsp, "view", è stato posto nella cartella WEB-INF, per impedirne l'accesso diretto da parte dell'utente nell'utilizzo dell'applicazione, così come è stata posta in WEB-INF la directory personalization, contenente gli script Python ed i file necessari all'interrogazione ed all'allenamento dei modelli di machine learning utilizzati nella componente di personalizzazione del sistema. Tale directory non è da confondersi con il package personalization, che contiene le classi java responsabili della componente di personalizzazione.

3. Class Interfaces

È possibile reperire la documentazione relativa all'interfaccia pubblica delle varie classi al seguente link (link alla documentazione).

Le specifiche delle interfacce, essendo il sito in lingua inglese, sono state sviluppate nella stessa lingua, per una maggiore comprensibilità e leggibilità del codice anche da parte di altri sviluppatori che dovessero entrare in contatto con il progetto.

4. Class Diagram

Il class diagram riportato nella pagina successiva rappresenta l'intero sistema. Ogni arco all'interno di questo grafo rappresenta una dipendenza o un'associazione tra classi.

Si noti che sono state riportate solo le dipendenze interne ad ogni package, e le dipendenze tra i package model, controller e view, mentre quelle in dettaglio tra le classi sono state riportate sopra, nell'analisi dei package (capitolo 2).

Si può notare come, rispetto al diagramma delle classi della fase di analisi, lo schema sia evoluto, applicando una serie di significative trasformazioni:

- Le relazioni molti a molti sono state implementate come collezioni.
- Sono state aggiunte diverse classi, nel passaggio dagli oggetti rappresentazione del dominio agli oggetti soluzione, quali:
  - RecommendedProductList, RecommendedProductListBuilder, PersonalizationBridge e tutte le classi del package "personalization" sono stati aggiunte per implementare la lista dei prodotti consigliati seguendo il design pattern Proxy, e per permettere la comunicazione tra gli script Python che implementano i modelli di Machine Learning. In particolare RecommendedProductList è stata creata per implementare la relazione molti a molti di personalizzaione.
  - L'intero package "dao", contenente i data access objects, è stato aggiunto, per fare da "interfaccia", seguendo il design pattern Data Mapper, con il database ed i dati persistenti;
  - L'intero package "controller", contenente le servlet, è stato aggiunto per implementare gli oggetti controller, già specificati nella fase di analisi;
  - L'intero package "view" è stato aggiunto per implementare gli oggetti boundary, già specificati nella fase di analisi.

(Link all'immagine)

5. Glossario

Componenti off-the-shelf: Componenti software sviluppati da terzi riutilizzabili.

JavaScript: Linguaggio di scripting utilizzato per dare dinamicità alle pagine html.

CSS: Linguaggio di formattazione per lo stile delle pagine html.

JSP: Tecnologia che facilita lo sviluppo di pagine web dinamiche.

Javadoc: Documentazione generata a partire da particolari commenti scritti all'interno di una classe java.

Java: Linguaggio di programmazione orientato agli oggetti.

Python: Linguaggio di programmazione orientato agli oggetti di alto livello.

Jep: È un'implementazione "embedded" di un compilatore CPython in Java, che sfrutta JNI.

JNI: acronimo di Java Naming Interface, è il servizio di naming di java, che permette ad un provider di registrare dei servizi su un registry, associandoli ad un nome, attraverso il quale poi i client, effettuando un'operazione di lookup, possono accedere.

Pandas: È una libreria Python per l'analisi e la manipolazione dei dati.

CSV: È un formato di file basato su file di testo utilizzato per l'importazione ed esportazione di una tabella di dati.

6. Riferimenti

- jQuery
- JUnit
- MySQL
- Selenium
- Mockito
- Spring
- Jep
- OpenCSV
- Scikit-learn
- Pandas
- Pickle
- Bootstrap
- Jstl
- Python
- RAD (Requirements Analysis Document) par. 3.5.3.2, modello a oggetti della fase di analisi.