Università degli Studi di Salerno
Corso di Ingegneria del Software

JapanWorld
Objective Design Document
Versione 1.1

Data: 17/08/2020

Progetto: JapanWorld	Versione: 1.1
Documento: Objective Design Document	Data: 25/08/2020

Coordinatore del progetto:

Nome	Matricola
Emanuele Patella	0512104730

Partecipanti:

Nome	Matricola
Emanuele Patella	0512104730

Scritto da:	Emanuele Patella

Revision History

Data	Versione	Descrizione	Autore
20/08/2020	1.0	Objective Design Document Versione 1	Emanuele Patella
25/08/2020	1.1	Revisione Documento	Emanuele Patella

Indice

1. Introduzione

1.1. Obiettivi di design e trade-off

Performance, Sicurezza, Usabilità e Robustezza, sono questi gli aspetti che si è deciso di privilegiare nello viluppo di questo e-commerce.

Proprio per questo motivo lo sviluppo del sito si è concentrato sul rendere la piattaforma che stiamo andando a creare facile da utilizzare, accattivante, rapida nel rispondere alle richieste e molto guidata nelle operazioni, riducendo così al minimo la possibilità di esiti indesiderati da parte di operazioni strane da parte dell'utente.

In funzione di ciò l'architettura e la divisione in Package del sito ne ha risentito, dividendo l'intero sistema in 3 soli package, corrispondenti alle 3 componenti del modello MVC citato nei documenti precedenti.

Questo documento prenderà in analisi tutti i package e le singole classi, andando a spiegare come interagiscono le une con le altre al fine di far funzionare la piattaforma in sviluppo.

1.2. Linee guida per l'utilizzo dell'interfaccia

I nomi di tutte le classi avranno la prima lettera maiuscola e vi saranno successive maiuscole solo e soltanto se il nome del documento contiene più di una singola parola.

I metodi avranno tutti nominativi che iniziano per lettera maiuscola e rispetteranno la stessa formattazione dei nominativi dei documenti in quanto a maiuscole/minuscole. Tendenzialmente saranno verbi all'infinito o brevi frasi.

Tutti i nomi saranno significativi e indicativi della funzione dei metodi e delle classi.

I file che andranno a comporre l'interfaccia saranno nominati e suddivisi nel modo seguente:

- I file che andranno a comporre l'interfaccia grafica avranno nomi composti da sostantivi che fanno riferimento all'attività la cui pagina è legata(es.: L'interfaccia della pagina di Login si chiamerà: "Login.html" ecc.);

- I nominativi dei controller, cioè, dei file che si occupano di processare i dati prima di passarli ai model per la query al database, saranno composti da brevi frasi che fanno riferimento all'operazione che la loro query svolgerà, quindi sono contraddistinti da verbi(es.: La funzione che processa i dati e permette di acquistare i prodotti presenti nel proprio carrello prende il nome di: "Acquisto.java");

- I nominativi dei model, infine, saranno dei sostantivi e fanno riferimento all'entità del database con cui andranno a interfacciarsi per effettuare la query(es.: Il model nel quale sono racchiuse le funzioni che interagiscono con i prodotti da acquistare si chiamerà: "Carrello.java")

1.3. Definitions, Acronyms and abbreviation

Non sono presenti acronimi o definizioni degne di nota.

1.4. References

Il sistema che andremo ad implementare, come già richiesto dal cliente nel proble statement, Dovrà avere una interfaccia estremamente intuitiva, ma accattivante, a questo scopo, all'interno del RAD, sono stati inseriti i mock-up per dare al cliente un'idea di come sarà l'interfaccia finale.

È stata anche assegnata una palette di colori da parte del committente e della quale si terrà conto nella fase di implementazione dell'interfaccia grafica del sito.

All'interno dell'SDD sono stati definiti i 3 package in cui il sito sarà diviso e le funzionalità che dovranno esporre, il tutto verrà esteso e spiegato al meglio all'interno di questo documento.

2. Pacchetti

2.1. Pacchetti progetto

La decomposizione del progetto in pacchetti è sostanzialmente composta di tre macro-blocchi che interagiscono tra di loro, come l'architettura MVC e come indicato nel documento precedente.

I tre Layer che andranno a comporre il progetto sono:

- Il Presentation Layer(View), che si occuperà di gestire tutte le componenti grafiche del progetto e permettere all'utente di comunicare con gli altri due Layer attraverso i comandi creati appositamente.
- Il Business Layer(Model), che si occuperà dell'acquisizione dei dati inseriti dall'utente e della loro corretta formattazione per eseguire le operazioni richieste, interagendo con lo Storage Layer ove necessario.
- Lo Storage Layer(Controller), che si occuperà di gestire tutte le interazioni con il database, gestendo le query che verranno fatte al sistema e formattando in maniera corretta i dati che il model andrà a leggere e, a sua volta, formatterà in maniera coerente alla visualizzazione richiesta dal Presentation Layer.

2.1.1. Package Model

Il Package model è quello preposto a far comunicare la View e il controller, attraverso la ricezione e la apposita formattazione dei dati. Il compito principale del model è quello di essere un ponte tra le due componenti più "attive" del sistema. Questo Package dovrà occuparsi di acquisire i dati che verranno inseriti tramite l'interfaccia e formattarli nel modo adeguato per l'operazione che l'utente richiederà di compiere.

Questo package conterrà:

- Package Connettitore:
    - Classe "Connettitore": Questo metodo è un doGet() che prende come parametri in input la 'HttpServletRequest' e 'HttpServletResponse' per prendere in input il parametro

HttpSession per poter creare la connessione con il database del sito. Questo metodo serve a passare i dati al metodo che creerà materialmente la connessione, all'interno del controller. Tale metodo non ha valori di ritorno. Viene richiamato dalla Home Page del sito all'avvio della sessione, di modo da permettere la comunicazione tra Controller e View.

- Package GestioneCarrello:
    - Classe "Acquisto": Altro metodo DoGet() con parametri 'HttpServletRequest' e 'HttpServletResponse' che si occupa di prendere i singoli prodotti ottenuti tramite la ricerca e aggiungerli al Carrello. Il metodo non ha valore di ritorno, come tutti i DoGet() presenti nel software. Viene richiamato da un bottone nell'interfaccia presente nella pagina di acquisto che gli fornisce, all'interno della request, l'identificativo del prodotto scelto.
    - Classe "Fattura": Metodo DoGet() privo di valori di ritorno e che prende in input, come i precedenti, le HttpServlet Request e Response. La funzione eseguita da questo metodo è la costruzione di alcuni parametri della fattura, insieme alla sua formattazione. Prende i parametri in input dall'interfaccia che si occupa di acquisire i dati per il pagamento e li invia alla classe "Pagamento", preposta ad inserirli nella relativa tabella del database.
    - Classe "Pagamento": Altro metodo DoGet() che, prendendo in input HttpServlet Request e Response, si occupa di inviare gli update al database relativi alle pratiche del pagamento. I metodo non ha valore di ritorno e, una volta completato l'update, reindirizza a una diversa View del sito, quella relativa allo scontrino. Questo metodo si occupa anche di caricare nel database i dati relativi alla fatturazione. Prende i dati, opportunamente formattati dalla Classe "Fattura" e li invia a un metodo del controller preposto al loro inserimento nella tabella apposita del database.
    - Classe "Rimozione": Metodo DoGet(), che prende in input i parametri HttpServlet Request e Response, il cui compito è rimuovere dal carrello i prodotti che l'utente non vuole più acquistare e riaggiungerli al database per rimetterli in disponibilità. Anche questo metodo non ha valori di ritorno ne reindirizzamenti ad altre pagine, poiché riaggiorna sempre la

stessa. Anche questa funzione, come nel metodo DoGet() di "Acquisto", riceve in input l'identificativo del prodotto da rimuovere tramite un bottone nell'interfaccia carrello, posto di fianco al prodotto, e passa questo specifico dato al layer dei Controller perché rimuovano il prodotto dal carrello e lo ri-aggiungano al database dei prodotti in disponibilità.

- Classe "StampaFat": Metodo DoGet(), che prende in input i parametri HttpServlet Request e Response, il cui scopo è quello di prendere dal database la fattura dell'utente e formattarla nel modo corretto per mostrarla a schermo. Il metodo in sé si occupa solo di trasmettere al controller Carrello i parametri necessari per scegliere la fattura giusta da visualizzare e caricarli nella Response a operazione compiuta. Anche questo metodo, come tutti gli altri, è privo di valori di ritorno.

- Classe "VisualCarrello": Metodo DoGet() con parametri in ingresso HttpServlet Request e Response e nessun parametro di ritorno. Lo scopo di questo metodo è richiamare un metodo del controller Carrello per ottenere tutti i valori dei prodotti presenti nel carrello e permetterne la corretta formattazione e visualizzazione a schermo. Questo metodo viene richiamato dall'interfaccia carrello presente nella View, e fornisce alla classe visual carrello, attraverso la Request, la stringa "carrello" presente in sessione. Il metodo richiamerà a sua volta una classe del controller preposta a effettuare una query al database per ottenere tutti i parametri necessari alla visualizzazione.

- Classe "VisualFattura": Metodo DoGet() gemello del precedente per valori in input ed i output. Lo scopo di questo metodo è di richiamare quello presente nel controller Carrello preposto alla formattazione e visualizzazione dei dati. Come sopra citato, questo metodo viene richiamato da una interfaccia preposta alla visualizzazione della fattura affinché possa ricevere, nella Response da parte del metodo, i dati da mostrare a schermo, correttamente formattati e ordinati.

- Package GestoneProdottiAcquistabili:

  - Classe "AggiungiProdotto": Metodo DoGet() che prende in input i parametri HttpServlet Request e Response e non ha valori di ritorno. La sua funzione all'interno del codice è quella di permettere di inserire nuovi prodotti all'interno del database,

permettendo anche solo di aumentare le quantità di quelli già presenti. Anche questo metodo, come tutti gli altri, ha solo la funzione di prendere i parametri in input e fornirli all'apposito controller per eseguire l'operazione sui dati presenti nel database. Viene richiamato da un bottone presente nella schermata di visualizzazione dei prodotti e che gli fornisce l'identificativo del prodotto selezionato. Nel caso di nuovo prodotto, cioè un prodotto non ancor presente nel database, il bottone invia un valore speciale per segnalarlo.

- Classe "Deposito": Metodo DoGet() che prende in input HttpServlet Request e Response per permettere al proprietario del sito la visualizzazione dei prodotti ancora presenti nel database con relative quantità. Il metodo non ha valore di ritorno ed è necessario per avere un feedback grafico delle azioni effettuate dal metodo "AggiungiProdotto" precedentemente citato. Il metodo viene richiamato dall'interfaccia, sopra citata, nella quale è anche presente il bottone che interagisce con la classe "AggiungiProdotto".

- Classe "RicercaManga": Metodo DoGet() che prende in input i parametri HttpServlet Request e Response, di modo da ricavarne i filtri necessari alla formattazione della query di ricerca dei prodotti. A differenza dei precedenti questo metodo ha un richiamo a 2 metodi del controller Prodotti, uno per la ricerca tramite i filtri forniti dal sito, e uno per la ricerca tramite parola chiave fornita dall'utente. Questa scelta è gestita da un if() che verifica se uno dei due parametri è vuoto. Anche questa funzione non ha valore di ritorno. Il metodo viene richiamato dai pulsanti presenti sulla barra di ricerca del sito e reindirizza alla pagina di visualizzazione prodotti, a prescindere dalla pagina di partenza.

- Package GestioneProfiliUtente:

  - Classe "CronologiaAcquisti": Metodo DoGet() che prende in input i parametri HttpServlet Request e Response. La funzione svolta da questo metodo è molto semplice, al click di un tasto sull'interfaccia dell'admin, visualizza tutti gli ordini attualmente in corso e quelli conclusi. Questa funzione serve principalmente a fini di contabilità e tracciabilità della merce. Tale metodo non ha valore di ritorno. Il metodo, come già detto, è richiamato dal click su di un tasto nell'interfaccia utente dell'admin, da non

confondere con l'interfaccia utente degli utenti generici, che ha un metodo a se stante.

- Classe "CronologiaAcquistiUtente": Metodo DoGet() Identico al precedente, se non per la particolarità di mostrare solo e unicamente gli ordini dell'utente che l'admin ha selezionato. La funzionalità principale, in questo caso, è la tracciabilità dell'ordine, oltre a rendere visibili eventuali operazioni non effettuate dall'utente in questione, oltre a permettere all'amministratore del sito di trovare più facilmente i dati di un ordine qualora dovessero insorgere problemi. Anche questo metodo prende in input i parametri HttpServlet Request e Response e non ha valori di ritorno.

- Classe "Iscrizione": Metodo DoGet() che prende in input i parametri HttpServlet Request e Response e ne ricava 3 variabili, e-mail, username e password. Questo metodo si occupa sia della fase di Login che di quella di registrazione, quindi esegue una serie di controlli[if()…else] per verificare se l'operazione che si sta eseguendo sia il login o la registrazione. In caso di registrazione, la e-mail non è nulla, quindi si procede a verificare che i dati non siano già presenti nel database e, in caso affermativo, si procede con la registrazione. Se l'e-mail è nulla, invece, si procede a verificare che lo username e la password corrispondano a quelli presenti nel database. In caso corrispondano a quelli di un utente generico, si viene reindirizzati sulla propria pagina personale, qualora corrispondessero, invece, alle credenziali dell'admin, si verrà reindirizzati sulla pagina dell'admin. Anche questo metodo è privo di valori di ritorno. Il metodo prende i parametri in input dalle due interfacce, Login e Registrazione, presenti nella View e reindirizza in schermate diverse a seconda dei risultati delle query effettuate dai controller associati.

- Classe "Logout": Metodo DoGet() che prende in input i parametri HttpServlet Request e Response che si occupa solo e unicamente di disconnettere il profilo di un qualsiasi utente, admin o non che sia, quando quest'ultimo ne fa richiesta. Il metodo non ha valori di ritorno. Il metodo è richiamato da un bottone presente nell'interfaccia delle aree utente.

- Classe "ModificaPassword": Metodo DoGet() che prende in input

i parametri HttpServlet Request e Response dai quali ricava due variabili: vecchiaPassword e nuovaPassword. L'operazione compiuta dal metodo subito dopo è abbastanza semplice, verifica che la vecchia password sia quella autentica, dopo di ché, se il controllo da risultato affermativo, la sostituisce con la nuovaPassword. Il processo tiene anche conto del fatto che la nuovaPassword rispetti le caratteristiche richieste da sito, cioè la sua alfa-numericità. Questo metodo non ha valore di ritorno. Questa funzione riceve i suoi parametri in input da una form all'interno dell'area utente.

- Classe "ModificaUsername": Metodo DoGet() che prende in input i parametri HttpServlet Request e Response dai quali ricava 3 parametri: vecchioUsername, Password e nuovoUsername. In questo caso vengono richiesti sia il vecchio username che la password per motivi di sicurezza. Dopo il check dei dati, si sostituisce il vecchioUsername con il nuovoUsername. Anche questo metodo non ha valore di ritorno. Questa funzione riceve i suoi parametri in input da una form all'interno dell'area utente.

- Classe "StatoPagamenti": Metodo DoGet() che prende in input i parametri HttpServlet Request e Response. Questa funzione prende lo username dell'utente loggato tramite i dati della sessione e gli permette di visualizzare tutte le sue transazioni, attuali e precedenti. Le funzionalità principali di questa funzione sono la verifica delle transazioni effettuate da un dato utente e la tracciabilità delle operazioni effettuate dal venditore, di modo da mantenere la transazione il più sicura e, allo stesso tempo, trasparente possibile. Il metodo viene avviato dal click su un apposito tasto all'interno dell'area utente.

2.1.2. **Package Controller**

Il package controller è quello preposto alle interazioni con il database, implementato tramite MySQL. Al suo interno saranno presenti tutte le classi preposte all'effettuare query o all'aggiornare il database.

Questo package avrà una corrispondenza biunivoca quasi totale con i file presenti all'interno del model, in quanto tutte le azioni dei controller sono innescate da azioni dell'utente sulla View e recepite tramite i model.

Questo Package conterrà:

- Package Carrello:
    - Classe Carrello:
        - Metodo VisualizzaFattura() è un metodo che prende in input due parametri, il parametro String utente e la sessione corrente, quindi HttpSession session. Il metodo sfrutta una serie di variabili locali per istanziare una data(Quella del giorno di fatturazione) e i vari dati necessari per la fattura(Codice Fiscale del negozio, ecc.). Dopo di ché scorre la stringa che contiene i dati del carrello della sessione attuale, per poi inserirli nella fattura e permetterne sia la registrazione nel database che la successiva lettura a schermo. Questo metodo restituisce un parametro Stringa che viene poi letto e decomposto per creare la fattura a schermo. Viene richiamato dal model "VisualFattura", che gli fornisce tutti i parametri necessari all'avvio della query.
        - Metodo InserisciProdottoCarrello() è un metodo che prende in input due parametri Stringa, cioè Utente e CodiceId. Lo scopo di questo metodo, attraverso lo sfruttamento di alcune variabili locali necessarie a tenere traccia di dati che non sono sempre necessari, quali la data e l'ora dell'acquisto, è quello di inserire i singoli prodotti all'interno della tabella del database preposta a tenere traccia degli acquisti effettuati dall'utente. Il metodo, oltre che dalla solite sezione in cui vengono instaziate le varie variabili locali, è composto da un ciclo che effettua la query al database e verifica se il prodotto sia già stato inserito(nel caso di multiple copie dello stesso prodotto ne viene aumentata la quantità), o, altrimenti, lo aggiunge alla tabella con quantità 1. Il metodo ha come valore di ritorno un Float, che va a indicare, alla funzione chiamante, il totale speso dall'utente, di modo da poterlo inserire come parametro all'interno della fattura.
        - Metodo Acquisto() è un metodo che prende in input 3 parametri di tipo Stringa, cioè nomeProdotto, genere e text. I primi due parametri, cioè nomeProdotto e genere servono per effettuare la ricerca del prodotto che l'utente vuole aggiungere al carrello, mentre il terzo, cioè text, è letteralmente la stringa che contiene tutti i prodotti presenti

nel carrello fino a questo punto. La stringa text viene mantenuta nella sessione e permette la gestione di multipli carrelli su multiple sessioni senza costringere il sistema a coinvolgere il database nell'operazione di persistenza dei dati. Il metodo si compone di un update al database, nel quale la quantità del prodotto scelto viene ridotta di una unità e poi l'Id del prodotto viene inserito nella stringa text, che viene data come valore di ritorno alla funzione chiamante. Il metodo viene richiamato dal model associato al bottone nell'interfaccia di ricerca dei prodotti.

- Metodo Visualizza() è un metodo senza parametri in ingresso che serve a visualizzare tutti i prodotti che sono stati venduti, fino a questo momento, dal sito, con relativi prezzi e fatture. Il metodo restituisce un valore di tipo stringa, che non è altri che l'insieme dei parametri che il metodo chiamante necessita per permettere all'interfaccia di visualizzare i vari prodotti. Questo metodo viene usato dall'area utente dell'admin, per permettergli di visualizzare a contabilità del proprio sito.

- Metodo VisualizzaUtente() è una funzione che prende in input un solo parametro stringa, cioè utente. Questa funzione è sostanzialmente identica alla precedente, ma mostra le fatture del singolo utente all'interno del proprio profilo. Il valore di ritorno, come per il precedente, è una stringa con tutti i dati necessari alla visualizzazione. Questo metodo viene utilizzata dall'area utente dei singoli iscritti alla piattaforma per poter visualizzare i loro acquisti fatti e in corso.

- Package <u>Connector:</u>
  - Classe DriverManagerConnectionPool:
    - Metodo CreateDBConnection() è un metodo senza parametri in input che permette al sistema di generare la stringa d'accesso necessaria a istanziare la connessione con il database. Il metodo può lanciare eccezioni, nello specifico la SQLException per segnalare eventuali problemi nell'istanziare la connessione. Il valore di ritorno di questa funzione è il parametro Connection, necessario per prendere possesso del canale di comunicazione tra

programma e database. Il metodo è richiamato dal model "Connettitore" al momento dell'avvio del sito per avviare la connessione.

- Metodo getConnection() è un metodo senza parametri in ingresso che permette alla funzione richiedente di ottenere il permesso di comunicare con il database. Il metodo, come tutti quelli della funzione DriverManagerConnectionPool, può lanciare la SQLException, qualora il tentativo di acquisizione della connessione non andasse a buon fine. Il valore di ritorno è il parametro Connection. Viene utilizzato da tutti i model che necessitano di richiamare il database per le loro interazioni.

- Metodo releaseConnection() è un metodo che prende in input il parametro Connection e ha la funzione di interrompere completamente la connessione della pagina con il database. Questo metodo è necessario per evitare che rimangano attive sessioni anche quando gli utenti hanno completato tutte le loro operazioni e hanno chiuso la pagina. Il metodo non ha valore di ritorno ma può, come gli altri, lanciare la SQLException per indicare eventuali problemi al sistema.

- Package <u>ProfiliUtente:</u>
  - Funzione Login:
    - Metodo Loggati() è un metodo con due parametri Stringa in input, Username e Password, che verifica queste credenziali per permettere l'accesso alla propria area personale agli utenti. Il metodo consiste in una query che viene utilizzata per vedere se queste credenziali sono nel database, se sono presenti, invia una stringa indietro che corrisponde ad "admin", se l'utente è l'admin del sito, a "utente" se è un utente generico, o a "noUtente" se l'utente non è registrato. Questo metodo viene richiamato dal model connesso alla interfaccia di Login.
    - Metodo Iscrivimi() è un metodo con tre parametri Stringa in input, Username, Password e Email. Questo metodo svolge solo due istruzioni, prima effettua una query al database, di modo da verificare che l'utente non sia già registrato. In

caso di nuovo utente, effettua l'update del database aggiungendolo al database. Il metodo ha poi un valore di ritorno di tipo Booleano, cioè, ritorna true(vero) se è tutto stato eseguito correttamente e l'utente è registrato, false(falso) altrimenti. Questo metodo viene richiamato dal model che interagisce con la pagina di iscrizione al sito.

        - Metodo Logout() è un metodo con un solo parametro in input di tipo Stringa, cioè username. Questo metodo serve a cambiare uno dei valori all'interno della tabella, cioè la colonna "connesso". Questo metodo setta il valore a 0(non connesso) quando l'utente si disconnette. I precedenti metodi lo settano ad 1(connesso) quando gli utenti loggano nei loro profili. La presenza di questo parametro serve solo a fini statistici, in quanto l'admin potrebbe usare questo dato per calcolare, con più facilità, il numero medio di utenti connessi contemporaneamente. Tale funzione non è stata inserita poiché non richiesta durante la creazione del sito, ma ho ritenuto opportuno creare questa colonna nel database per avvantaggiarmi qualora venisse chiesta un'aggiunta di questo tipo al sito. Il metodo non ha valore di ritorno e può lanciare la SQLExeption, proprio come i precedenti. Questo metodo viene richiamato dalle varie pagine dell'area utente, sia Admin che non.

    - Classe "Utente":

        - Metodo CronologiaAcquisti() è un metodo che prende in input un parametro Stringa, username, e lo sfrutta per mostrare all'utente la cronologia dei propri acquisti. Il metodo ritorna un parametro Stringa, che altri non è che l'elenco dei dati che devono essere poi formattati dal model per essere mostrati poi a schermo. Il metodo consiste solo nella query al database e al settaggio della Stringa che verrà poi restituita come risultato. Questo viene richiamato dalle pagine di area utente, Admin e non, per permettere la visualizzazione dei prodotti acquistati.

- Package <u>Prodotti:</u>

    - Classe "Prodotti":

        - Metodo Ricerca() è un metodo con un solo parametro

Stringa in input, cioè il parametro che farà da filtro per la ricerca. Questo metodo è la funzione che si occupa di gestire la ricerca tramite filtro inserito dall'utente, nella barra di ricerca. Questo metodo esegue una semplice query al database e mostra tutti i risultati che includono il filtro inserito dall'utente come serie o genere(Questa è la differenza sostanziale dalla query successiva). Il valore di ritorno è una Stringa con all'interno tutti i risultati della query. Utilizzato dal model connesso alla view di ricerca per ottenere i dati da mostrare a schermo.

- Metodo RicercaProdotti() è un metodo con due parametri Stringa in input, serie e genere. La sua funzione è la stessa del metodo precedente, l'unica differenza consiste nel fatto che qui il parametro di ricerca non è quello immesso dall'utente nell'apposita barra ma si fa, invece, affidamento ai filtri forniti dal sito, che classificano ogni prodotto in base a due parametri(serie e genere, per l'appunto) e non secondo un parametro unico. L'unica altra differenza dalla funzione precedente consiste nel fatto che la query al database usa un AND e non un OR nella definizione di filtri. Il valore di ritorno è di tipo Stringa e consiste nell'elenco dei parametri da visualizzare. Utilizzato dal model sopra citato.

- Metodo RicercaProdottiCarrello() è un metodo con un solo parametro Stringa in input, di nome CodiceId. La funzione di questo metodo è quella di prendere in input i CodiceId(Chiave per la tabella prodotti che distingue univocamente tutti i prodotti nel magazzino) di tutti i prodotti inseriti nel carrello per ottenere, tramite la query al database, tutte le altre componenti dei prodotti, poiché richiederebbe una quantità di spazio e uno sforzo inutile memorizzare, per ogni elemento inserito nel carrello, tutte le componenti che lo identificano. La funzione restituisce una Stringa con tutte le caratteristiche che verranno elencate a schermo dalla funzione chiamante. Utilizzato dal model connesso all'interfaccia di visualizzazione del carrello.

- Metodo RicercaProdottiAd() è un metodo che prende in input due parametri Stringa, serie e genere. Questa

funzione è quella che viene richiamata quando l'utente fa una ricerca per decidere quali prodotti acquistare utilizzando i filtri forniti dal sito. La differenza sostanziale che questo metodo ha dal metodo RicercaProdotti() è che questo metodo visualizza tutti i prodotti, anche quelli la cui quantità è 0, poiché il metodo è utilizzato dalla pagina dell'admin per mostrare i prodotti che sono presenti nel deposito e, eventualmente, aggiornarli attraverso gli appositi metodi. Anche questa funzione restituisce un parametro Stringa corrispondente all'elenco delle componenti da mostrare a schermo.

- Metodo addProdotti() è un metodo che prende in input 7 parametri, String CodiceId, String NomeProdotto, String genere, String serie, int prezzo, int quantita, String immagine. Questo metodo, come intuibile dai parametri in input, si occupa di aggiungere prodotti al sito, qualora non vi fossero già stati inseriti. Il metodo non ha valore di ritorno ma può lanciare SQLException, qualora l'inserimento avesse dei problemi. Viene utilizzato da un model connesso alla pagina dell'admin per permettere l'aggiunta di nuovi prodotti al database.

- Metodo modificaProdotti() è un metodo con un solo parametro in input di tipo Stringa, cioè CodiceId. Permette di aumentare di uno la quantità del prodotto di cui si inserisce il CodiceId. Nessun valore di ritorno ma possibilità di lanciare la SQLException in caso di problemi con l'update. Richiamato dallo stesso metodo sopra-citato.

- Metodo removeProdotti() è un metodo con un solo parametro in input, String CodiceId che permette di rimuovere dall'interno del database il prodotto il cui CodiceId viene inserito nel database. Anche qui, nessun valore di ritorno ma possibilità di lacio di SQLException da parte del metodo. Richiamato dal model connesso all'interfaccia admin preposto all'interazione con i prodotti nel database.

2.1.3. **Web Content(Package View)**

Il Package Web Content è la View della nostra architettura MVC. È composto

da numerosi file che vanno a comporre tutta l'interfaccia del nostro sito. È in assoluto la componente più importante delle tre descritte fino ad ora, poiché, senza l'interfaccia, gli utenti non potrebbero comunicare col sistema. La struttura dell'interfaccia serve anche a limitare le azioni degli utenti, di modo da evitare eventuali problemi di input e situazioni spiacevoli dovute a azioni non coerenti col sistema che stiamo sviluppando.

In tutte le View sarà presente un elemento comune che descrivo qui, di modo da non ripetermi all'interno della descrizione delle funzionalità. Tutte le View avranno una barra, presente nella parte alta dello schermo, che manterrà disponibili tutte le funzioni all'utente, incluso il bottone di Logout, qualora fosse connesso con il proprio profilo.

Questo Package conterrà:

- AdminControlPage: Pagina nella quale l'admin del sito può gestire tutti i dati relativi alle vendite effettuate e le quantità di prodotti presenti all'interno del magazzino, di modo da mantenere le scorte sempre disponibili. L'interfaccia consta di vari bottoni che permettono l'interazione con i quantitativi di prodotto presenti nel magazzino. Vi è, inoltre, un pulsante per la visualizzazione degli ordini con il relativo stato(in corso, consegnato, annullato). Viene richiamata da un bottone presente nell'interfaccia della AdminPage.

- AdminPage: Pagina principale che viene caricata quando l'admin accede alla propria area personale. Permette di raggiungere le funzionalità indicate nell'interfaccia precedente, oltre a permettere di modificare la propria password e di aggiungere e aggiornare tutti i contatti e i recapiti del sito. Questa pagina può essere raggiunta o dall'area login usando le apposite credenziali, oppure da un bottone sulla barra di navigazione una volta che si è effettuato il login.

- Assistenza: Interfaccia necessaria in tutti quei casi in cui si verifica un imprevisto nell'acquisto di un prodotto ed è quindi necessario contattare il rivenditore. Fornisce sia una chat diretta con il rivenditore, avviando le pratiche necessarie alla risoluzione del problema che gli altri contatti del negoziante, come: Profili social del negozio, numero di telefono, indirizzo mail ecc. Raggiungibile attraverso un apposito bottone nella barra di navigazione.

- CampiPagamento: Semplice pagina con una form necessaria a prendere i dati dell'utente durante il completamento di un ordine. È possibile accedervi solo tramite la procedura di pagamento e disabilita tutti gli

altri tipi di input dati, rimangono comunque attivi i filtri ricerca e il tasto home, qualora non si volesse portare a compimento l'operazione.

- Carrello: Interfaccia che mostra tutti i prodotti che l'utente ha messo da parte per poterli acquistare. È raggiungibile tramite una apposita icona presente nell'interfaccia e sfrutta i model e i controller per formattare correttamente i dati presenti nella session e visualizzarli a schermo, per permettere all'utente di controllare i prodotti e di rimuovere quelli che non ha più intenzione di acquistare. È presente, in oltre, un bottone che avvia le pratiche necessarie al pagamento.

- CronologiaUtente: Interfaccia gemella di quella Admin dedicata alla visualizzazione dei prodotti acquistati dai vari utenti, in questo caso specifico, però, vengono visualizzati solo gli ordini dell'utente in questione, permettendo di avere uno storico dei propri acquisti. In una versione definitiva questa funzione mostrerebbe anche il link per eventuali codici di tracciabilità delle spedizioni ancora in corso. Il tutto è accedibile tramite bottone dall'interfaccia utente.

- HomePage: Interfaccia chiave della struttura del nostro sito, da qui passano tutti gli utenti, in quanto è la prima interfaccia ad essere caricata e visualizzata da loro. È composta da un carousell che mostrerà i prodotti più nuovi presenti nel nostro magazzino, oltre al logo del negozio e a tutti i filtri presenti nella barra posta in cima alla pagina, con filtri di ricerca, barra di ricerca per l'inserimento di un filtro personalizzato, login e carrello. Nel momento del login effettuato il bottone verrà sostituito dall'icona del profilo, la quale permetterà di accedere alla propria area personale. È la prima pagina che verrà mostrata nel momento in cui si avvierà il sistema.

- Login: Interfaccia che permette l'accesso alla propria area personale, consta di una form compilabile con due campi(username e password), di un bottone di conferma e di un link per andare alla pagina di registrazione qualora non ci si sia ancora iscritti. Raggiungibile tramite un bottone posto sulla barra di navigazione e che viene sostituito da un link alla propria area utente dopo aver effettuato il login.

- ModificaCredenziali: Pagina dell'area utente che permette la modifica sia della credenziale password che di quella username relativamente al proprio profilo. È stata creata perché mantenere questa funzione sempre in vista nella schermata iniziale dell'utente al login sembrava scomodo e fastidioso, infatti si è optato per inserirla in una sotto-area tutta sua. La

pagina consta di due form, una per lo username e una per la password, oltre a due bottoni, uno ciascuno, per confermare il cambiamento appena richiesto. È possibile raggiungere questa funzione tramite un apposito bottone dell'area utente e Admin.

- Registrazione: Pagina precedentemente citata in Login. Permette di inserire i propri dati e registrarsi al sito. È composta da una form con 3 campi(username, e-mail e password) più un bottone di conferma e un link alla pagina di Login, qualora si fosse cliccato per sbaglio su registrazione. Una volta compilati i campi e effettuata la registrazione si viene automaticamente reindirizzati alla pagina di login per effettuare l'accesso. Raggiungibile tramite link presente nella pagina di Login.

- RicercaManga: Pagina dinamica del sito, che visualizza tutti i prodotti che corrispondono ai filtri da noi selezionati. È composta da una tabella che mostra tutti i prodotti, compresi di foto-miniatura e un bottone per ogni prodotto che ne permette l'aggiunta al carrello. La pagina viene caricata nel momento in cui l'utente effettua una ricerca.

- Scontrino: Pagina place holder necessaria per dare una parvenza di completo all'operazione di acquisto. Funge da visualizzatore per il pagamento completato e le è stato dato un aspetto simile a quello di un vero e proprio scontrino, ma verrà modificata quando si avrà la disponibilità di un formato effettivamente valido. Viene visualizzata alla fine delle pratiche di pagamento.

- UserPage: Pagina principale dell'utente non admin. Contiene un elenco degli ordini in corso, una piccola form per cambiare l'immagine dell'account o altri dati personali inseriti nel sito e il bottone di logout. Viene raggiunta dall'utente una volta effettuato il login o successivamente tramite l'apposito bottone che compare nella barra di navigazione al posto di quello di login.

- Style: file contenente la formattazione dell'interfaccia grafica e la palette cromatica indicata dall'utente. Questo accorgimento permette l'inserimento di altre interfacce senza dover, ogni volta, reinserire questi parametri.

3. Design pattern

Non sono stati utilizzati design pattern specifici per questo progetto, poiché il codice, nonostante la lavorazione e la correzione di diversi bug e problemi strutturali, si basa su un progetto sviluppato per l'esame di Tecnologie Software per il Web(TSW) ove i design pattern non erano ancora argomento di studio fondante nella realizzazione del progetto.

Le operazioni principali effettuate per rendere più chiaro il progetto, all'infuori di commenti e bug-fixing, sono state operazioni i re-packaging necessarie a rendere coerente la struttura con il modello MVC.

Se si cercano similitudini con design pattern per applicazioni Web esistenti, la più calzante è quella con il modello JSP/2, il quale sfrutta l'utilizzo combinato di JSP, Java Bean e Servlet. L'utilizzo di un sistema simile all'architettura JSP/2 è stato, tuttavia, fortuito, quindi non sarebbe corretto dire che il design pattern sia effettivamente stato scelto.

L'utilizzo di un metodo di questo tipo è subito spiegato dal fatto che si volesse mantenere una forte separazione tra presentazione, modello e logica di controllo.