**DOMANDE:**

* ***Come viene gestito il carrello?***

Il primo controllo che viene effettuato è di prelevare l’attributo “user”. Se non si è user si viene reindirizzati alla pagina di home. Vengono prelevati i dati dell’user e vengono mostrati in una sezione dedicata.

Quando il documento è pronto mediante *DoRetrieveCart* vengono prelevati i videogiochi presenti nel carrello del database di quell’utente. Se c’è almeno un videogioco viene calcolato il prezzo totale e la data di spedizione.

Con il comando $("#span-price").html(totPrice.toFixed(2)); viene mostrato sullo schermo il prezzo.

Con il comando for(let prod of cart){

$("#column").append(generateCard());

}

Per ogni videogioco nel carrello verrà creata una card di informazione del prodotto.

Con il comando $(".fatherofall").each(**function**(){ aiuta a rappresentare le informazioni del videogioco.

Con il comando $("#btn-orderNow").click(**function**(){ si interroga il database (DoBuyCartVideogames) aggiungendo l’ordine appena effettuato e rimuove contemporaneamente i videogiochi nella tabella CARRELLO.

* ***Come viene fatto il login?***

In login.jsp, sono presenti gli input di username e password. Il file js-login.js verifica le validità dei campi. All’interno della jsp è presente un testo inizialmente nascosto che indica che la password non rispetta i requisiti. Js-login.js ne controlla la validità, se non è buono mostra il messaggio di errore. I seguenti dati verranno mandati a DoLogin che preleverà i 2 parametri e per prima cosa verifica l’esistenza di quell’utente. Verranno effettuati diversi controlli per il login, si verrà reindirizzati alla pagina index.jsp.

* ***Dove usi la responsiveness?***

La responsiveness permette di riadattare gli elementi della pagina html visualizzandoli correttamente su ogni dispositivo con schermi di dimensioni diverse. Abbiamo utilizzato un file css al cui interno è presente il comando @media. Ad esempio se lo schermo raggiunge determinati requisiti, ci saranno vari cambiamenti sulla visualizzazione degli elementi.

* ***Perché ho gestito il carrello con il database?***

Per semplicità e per non sovraccaricare la sessione.

* ***Dove e perché usi le chiamate ajax?***

Uso Ajax per richiedere dati/risorse in modo asincrono al server. Il metodo ajax utilizzato mediante jquery è: “get” che consente di eseguire prima una risorsa (nel nostro caso un metodo) e poi di eseguire una callback. Esempi nel nostro progetto sono:

**js-banpanel.js**

In questo caso, verrà prima eseguita la Servlet DoGetBannedUsers che ritornerà un ArrayList<UserBean> di utenti bannati. Dopo che la servlet è stata completata verrà eseguita la funzione successiva che genererà le card con gli utenti bannati in banpanel.jsp.

$.get("DoGetBannedUsers", function(data, status) {

var bannedUsers = JSON.parse(data);

// alert(bannedUsers);

for (let user of bannedUsers) {

$("body").append(generateUserCard());

}

var userName = btn.parent().prev().children().eq(0).html();

$.get("UndoUserBan?username=" + userName + "", function(data, status) {

// alert(data);

location.reload(false);

})

Viene prelevato l’username, mediante la get viene eseguita prima la Servlet UndoUserBan passando il parametro userName per sbannare l’utente (settando l’attributo canAccess == true). Viene poi eseguita la funzione che non fa altro che ricaricare la pagina.

**js-cart.js**

$.get("DoRetrieveCart",(data,status)=>{

var cart=JSON.parse(data);

if(cart.length>0){

………………….

}

Viene prima eseguita la Servlet DoRetrieveCart che mediante l’user preleva tutti i videogiochi che sono nel carrello (possono anche non esserci) con le relative informazioni. Viene poi eseguita la funzione che per ogni videogioco genera una card con le sue informazioni. Sono anche presenti il prezzo totale e le data di arrivo.

**js-index.js**

$.get("DoGetCategories",function(data,status){

//alert(data);

var dropD=$('#dropcateg');

var catVec=JSON.parse(data);

for(let cat of catVec){

var element=$('<li>');

var link=$('<a>');

link.attr("href","DoRetrieveByCategory?category="+cat);

link.html(cat);

element.append(link);

dropD.append(element);

}

})

Viene prima eseguita la Servlet DoGetCategories che preleva le Console dei videogiochi. Viene eseguita una funzione che crea una lista i cui elementi sono le console. Per ogni console è presente un href che chiama la Servlet DoRetrieveByCategory (con la console) che genera una pagina in cui si visualizzano tutti i videogiochi di quella console.

$.get("HomeSuggestions",function(data, status){

var suggestions = JSON.parse(data);

var index = 0;

$(".target").each(function(){

…………………………………

}

Viene prima eseguita la Servlet HomeSuggestions.java che genera 6 videogiochi scelti casualmente dal database. Viene poi eseguita la funzione che, per ogni videogioco, genera una card con le relative informazioni. Cliccando sull’immagine si aprirà la scheda del videogioco (getVideogameLink).

Viene prima eseguita la Servlet DoLogout.java che ti farà uscire dal tuo account sul sito e di conseguenza diventerai un visitatore. Viene poi eseguita la funzione che ricarica la pagina.

$.get("DoLogout",function(data, status){

window.location.reload(false);

return;

});

**VISUALIZZAZIONE DEGLI 8 GIOCHI NELLA HOME**

La classe HomeSuggestions.java crea un ArrayList di videogiochi (size: 8) passandola al metodo interno getJSONSuggestions che presi tutti i videogiochi ne rimuove “k” fintanto che non ne rimangono 8 (scelta casuale col metodo shuffle). Tramite un for vengono create 8 risorse ognuna delle quali contiene (“price, title, imgPath,VideogameCode”).

All’interno di index.jsp ci sarà la classe target, mediante questa il file js-index.js riempirà gli 8 campi della home.

**CRITERIO DI RAGGRUPPAMENTO CONSOLE**

Per raggruppare i giochi per console, nel momento in cui viene caricata index.jsp la classe DoGetCategories interroga il database (mediante DAO) e ottiene le 3 console.

Mediante js-index.js la linea di codice “$.get("DoGetCategories",**function**(data,status)” una volta prese le 3 categorie, aggiunge queste ultime alla lista di console. Ogni elemento ha il suo link che chiama DoRetrieveByCategory che permette di visualizzare i videogiochi della sua console.

searchresult.jsp ha il compito di mostrare i videogiochi inerenti alla console.

**BARRA DI RICERCA VIDEOGIOCO**

In index.jsp è presente un form all’interno del quale è presente un tipo input text con nome “searchBar”. Il seguente parametro verrà prelevato dal DoSearch.java che crea inizialmente un ArrayList di videogiochi vuoto e andrà ad aggiungere il videogioco. Viene fatto il dispatcher al searchresult.jsp che inizialmente controlla che se l’array è vuoto, verrà visualizzata che non ci sono videogiochi, altrimenti uscirà il videogioco.

**ORGANIZZAZIONE LOGIN & LOGOUT**

In index.jsp è presente un controllo se in un determinato momento che visualizzi la pagina, sei un user, admin o un visitatore. Se si è un visitatore, nella barra appariranno i bottoni di login e signup.

Se si è un user ci sarà la possibilità di vedere gli ordini effettuati (DoRetrieveOrder), vedere le impostazioni dell’account (./account.jsp), visualizzare il carrello(./cart.jsp) ed effettuare il logout (id: spanLogOut DOPO VEDI JAVASCRIPT).

Se sei un admin non puoi effettuare acquisti ma hai la possibilità di: Ban/Unban un user (./banpanel.jsp), aggiungere o rimuovere un admin (./adminpanel.jsp), aggiungere o rimuovere un videogioco (./videogamepanel.jsp), vedere le impostazioni dell’account (./account.jsp).

**SIGNUP (REGISTRAZIONE)**

In signup.jsp è presente un form che preleva tutti i dati di registrazione (un file javascript ne controlla la correttezza dell’ input.). Tutti questi dati verranno mandati a DoSignUp.java che controllerà ulteriormente la correttezza dei dati inseriti, verranno aggiunti ad un oggetto UserDAO per poi essere inseriti nel database. Viene anche settato un messaggio di conferma della registrazione in signup.jsp

**BAN/UNBAN DI UN USER**

È presente un input di testo che preso un user chiama la classe DoUserBan.java . In questa classe viene prelevato il parametro “username” e nell’oggetto UserBean viene prelevato dal database quell’utente. Vengono effettuati una serie di controlli, se passati, lo user viene bannato (settando il campo canAccess= false). Viene settato il messaggio di conferma di ban e mandato nuovamente alla jsp. Nella jsp, mediante javascript, viene prelevato il messaggio, se è presente viene stampato a schermo e poi viene eliminato dalla sessione.