RELAZIONE PROGETTO DI TWEB

Martina Xhepa

Matricola 891242

Università degli studi di Torino

Esame del 17/02/2023

TEMA DEL SITO

Il sito "MatterHorn" simula un e-commerce dedicato alla vendita di prodotti da scialpinismo, quali scarponi, caschi, sci ecc..

SEZIONI PRINCIPALI

Per poter accedere alle varie sezioni del sito gli utenti devono prima effettuare il login, gli utenti consumatori possono registrarsi nel caso non lo abbiano già fatto.

Pagina Login Utenti: ../html/userlogin.php

Pagina Registrazione Utenti: ../html/usersignup.php

Pagina Login Amministratore: ../html/adminlogin

L'utente consumatore può consultare i prodotti filtrandoli per categoria (Pagina **../html/category.php**) oppure può decidere di visualizzare l'intero shop (Pagina **../html/shop.**php). Per ogni prodotto può accedere ad una pagina ad esso dedicata (Pagina **../html/view.php**).

L'utente può decidere di aggiungere un prodotto al carrello oppure alla wishlist (può anche decidere di eliminarli). I prodotti aggiunti al carrello vengono visualizzati nella pagina **../html/cart.php**, quelli aggiunti alla wishist vengono visualizzati nella pagina **../html/wishlist.php**. Se un prodotto si trova nella wishlist e viene aggiunto al carrello, esso verrà eliminato dalla lista dei desideri.

L'utente amministratore possiede sulla navbar la voce "AdminPanel" che lo reindirizza alla dashboard (pagina **../html/admindashboard.php**), qui può visualizzare quanti prodotti ha, quanti utenti sono registrati nel sito e dalla pagina dei prodotti **(../html/products.php**) può visualizzare i prodotti, eliminarli o aggiungerne di nuovi.

FUNZIONALITA'

LOGIN UTENTE

Gestito nella pagina ../html/userlogin.php (la quale presenta anche un bottone che reindirizza alla registrazione e uno che reindirizza alla pagina login dedicata agli amministratori). I dati inseriti nel modulo vengono controllati lato client (../js/userlogin.js) con jQuery che restituisce il feedback "I valori inseriti non hanno un formato valido" nel caso in cui uno o più campi siano vuoti o se la mail

non presenta un formato valido. Se i dati sono validi vengono inviati tramite POST al file ..php/userlogin.php che si occupa della validazione lato server (interrogando il DB controlla che i dati siano corretti e coincidano con quelli registrati) e setta le sessioni. Viene restituito un file json, utilizzato da jQuery per fornire feedback all'utente: se il login va a buon fine l'utente viene reindirizzato alla homepage, altrimenti se il file json restituisce "error" viene visualizzato il messaggio "Email o password sbagliati".

LOGIN AMMINISTRATORE

Il discorso è simile a quello per l'utente, ma la pagina che lo gestisce è ../html/adminlogin e i dati vengono controllati e validati lato client e server rispettivamente da ../js/adminlogin.js e ../php/adminlogin.php

SIGNUP UTENTE

Gestito nella pagina ../html/usersign.php. I dati inseriti nel modulo vengono controllati lato client (../js/usersignup) che restituisce il feedback "I campi non possono essere vuoti" nel caso in cui uno o più campi siano vuoti o "L'email non ha un formato valido" se la mail non presenta un formato valido. Se i dati sono validi vengono inviati tramite POST al file ../php/usersignup.php che si occupa della validazione lato server (interrogando il DB controlla che lo username o la mail non siano già esistenti e registra il nuovo utente) e setta le sessioni. Viene restituito un file json, utilizzato per fornire feedback all'utente: se il login va a buon fine l'utente viene reindirizzato alla homepage, altrimenti se il file json restituisce "username" viene visualizzato il messaggio "Lo username è già utilizzato", se json restituisce "email" il feedback all'utente sarà "L'email è già registrata".

LOGOUT

Nella navbar troviamo una sezione Logout (che reindirizza a ../php/userlogout.php) che permette di effettuare la session_unset() e session_destroy() e poi reindirizza a ../html/userlogin.php.

CONTENUTO GENERATO DALL'UTENTE

L'utente può aggiungere prodotti al carrello o alla wishlist. Può poi visualizzare queste pagine e decidere di eliminare qualche prodotto o anche tutti. Se un prodotto si trova già nel carrello e l'utente prova ad aggiungerlo nella wishlist verrà notificato che il prodotto è presente nel carrello quindi non può essere aggiunto alla wishlist. Se invece dalla wishlist decide di aggiungere il prodotto al carrello, esso verrà inserito nel carrello ed eliminato dalla wishlist.
L'aggiunta di un prodotto al carrello comporta una insert nella tabella cart, l'aggiunta di un prodotto nella wishlist comporta una insert nella wishlist (l'eliminazione comporta dei delete).
L'utente amministratore può aggiungere ed eliminare prodotti, provocando rispettivamente una insert o una delete nella tabella product. Se elimina un prodotto e questo si trova all'interno della wishlist o del carrello di un utente, il prodotto verrà eliminato anche da queste pagine.

USABILITA'

Il sito è reso responsive tramite media queries per potersi adattare a diversi dispositivi.
Le azioni dell'utente presentano evidenti feedback visivi, per esempio nelle pagine di login e registrazione appaiono delle scritte colorate in rosso nel caso in cui i dati inseriti non siano idonei, per aiutare l'utente a capire cosa può aver sbagliato.

Vi sono animazioni anche quando per esempio l'utente passa con il mouse sopra ad un prodotto (compaiono due icone, una reindirizza alla pagina specifica del prodotto, l'altra lo aggiunge alla wishlist). Compaiono messaggi anche quando l'utente aggiunge un prodotto alla wishlist e al carrello o quando l'amministratore compila il form per l'aggiunta di un prodotto.

SESSIONI

Vengono aperte con il login (che deve andare a buon fine) e chiuse con il logout. Se l'utente è consumatore si salva la variabile 'userLoggedIn' e le variabili user_id (relativo all'id dello user nella tabella del db 'users') e user_name (relativo allo user_name inserito durante la registrazione). Se l'utente è amministratore si salva la variabile 'isLogged' e le variabili admin_id (relativo all'id della tabella 'admin') e admin_name (relativo al name nella tabella).

INTERROGAZIONI DEL DB

Le interrogazioni sono effettuate da diverse pagine del sito. Alcune effettuano query di tipo INSERT come usersignup.php che deve inserire i dati del nuovo utente registrato, oppure come shop.php e category.php che, nel caso l'utente aggiunga un prodotto alla wishlist o al carrello, devono aggiornare con il prodotto aggiunto le tabelle 'wishlist' o 'cart'. La pagina products.php è utilizzata dagli amministratori per aggiungere prodotti nuovi e si opera una query INSERT sulla tabella 'product'.
Product.php, cart.php e wishlist.php operano query DELETE nel caso l'utente elimini un prodotto e quindi devono aggiornare le tabelle.
La pagina shop.php richiede al database tutti i prodotti, category.php chiede tutti i prodotti di quella categoria, cart.php chiede i prodotti del carrello di un certo utente, whishlist pure.
Tutte le interrogazioni vengono gestite con l'utilizzo di JQuery e Ajax. Ogni chiamata ajax si riferisce ad un certo file.php che interroga il database e restituisce i dati in formato json, pronti per essere manipolati in jQuery per generare il contenuto richiesto dall'utente.

VALIDAZIONE DEI DATI LATO CLIENT E SERVER

I dati sono validati da entrambi i lati.
Esempio lato client: nella registrazione e nel login si usa regex per la mail e si controlla che i dati non siano nulli.
Esempio lato server: si controlla che la mail o lo user non siano già presenti nel database.

SICUREZZA

Per accedere a tutti i contenuti del sito si deve per forza passare per il login.
Quando un utente si registra la password viene cryptata con md5.
Le variabili passate per le query vengono messe nella funzione htmlspecialchars().
Si utilizzano oggetti PDO con prepare() ed execute() per impedire l'utilizzo di script melevoli.

PRESENTAZIONE

A parte la pagina di login e registrazione, le pagine presentano in alto uno header (con Nome del sito, icone per carrello e wishlist e la navbar con i link allo shop alla home e al logout), una parte centrale di contenuto e un footer (dove troviamo informazioni su contatti e social media).

Lo header della dashboard dell'amministratore presenta i link per il logout, per l'eccomerce e per la pagina che gestisce i prodotti.

FRONTEND

Le pagine che verranno visualizzate dall'utente si trovano tutte nella cartella html, tranne index.php. Nella cartella css troviamo raggruppati i fogli di stile. Tutti i file js sono nella cartella js e hanno generalmente lo stesso nome dei file in html. I file js chiamo i file php contenuti nella cartella php, che contiene anche il file db.php che ospita la maggior parte delle funzioni di interrogazione al db.
In image troviamo le immagini utilizzate per il sito. In uploade_img invece troviamo le immagini dei prodotti caricati.
La cartella components contiene file comuni come il footer, l'header oppure connect.php per connettersi al dv.

BACKEND

ARCHITETTURA GENERALE CLASSI/FUNZIONI PHP

La comunicazione tra la parte frontend e la parte backend è gestita con i file JS che inviano richieste AJAX ai file PHP i quali si occupano di interrogare il DB e restituiscono in output dati JSON. In base a questi poi di manipola il DOM per fornire all'utente il contenuto richiesto.

DATABASE

Il database shop è composto da 5 tabelle:
Users(ID, name, email, password);

Admins(ID, name, password);

Product(ID, name, details, price, image_01, image_02, image_03);

Wishlist(ID, user_id, pid, name, price, quantity, image);

Cart(ID, user_id, pid, name, price, quantity, image);

FUNZIONI REMOTE

Struttura generale:

-Il caricamento di una pagina o un evento comportano l'invio di una richiesta ajax da parte di '../js/file.js' a '../php/file.php', il quale opererà un interrogazione al DB.
-I dati vengono passati a '../php/file.php' tramite POST o GET, in base al caso.
-'../php/file.php' si occuperà di operare una SELECT, INSERT o DELETE in base a quello che serve e restituirà un file json.
-Il contenuto di questo file viene poi utilizzato lato javascript:, per generare contenuto in base al feedback json che ha ottenuto

Esempi generali:

-se la richiesta era di caricare tutto lo shop, se ci sono prodotti viene restituito un file json contenente i prodotti su cui con un forEach() vengono stampati tutti i prodotti. Se non ci sono prodotti json restituisce “empty” e l’utente visualizzerà un messaggio del tipo “Non ci sono prodotti caricati”

-se un amministratore elimina un prodotto dallo shop, viene inviata una richiesta ajax (dove si passa come dato il pid del prodotto tramite POST) , lato server si elimina il prodotto dalla tabella ‘prodotti’ e si eliminano anche i dati relativi a quel prodotto dalle tabelle ‘wishlist’ e ‘cart’

- …

Esempio specifico: usersignup

l’utente compila il form e i validi lato client sono validi

In ../js/usersignup.js:

Si preparano i dati da mandare a ../php/usersignup.php prendendo i valori inseriti negli input.

I dati vengono mandati tramite POST

```
$.ajax({
                url: '../php/usersignup.php',
                method: 'POST',
                data: {
                    signup: 1,
                    namePhp: name,
                    emailPhp: email,
                    passwordPhp: pwd
                },
```

In ../php/usersignup.php

Ottiene tramite POST i dati e cripta la password

```
$userEmail = htmlspecialchars($_POST["emailPhp"]);
    $password = htmlspecialchars($_POST["passwordPhp"]);
    $userName = htmlspecialchars($_POST["namePhp"]);
    $submitted_pw_hash = md5($password);
```

Si prepara l’interrogazione al database
Prima si controlla se l’email è già esistente, se lo è viene restituito sul file json “empty”,
se non lo è si controlla il nome. Se esiste viene restituito “username”
Se ne la mail ne il nome sono presenti viene effettuata una INSERTA del nuovo utente e restituisce “success”

```
if($select_user->rowCount()>0){
        echo json_encode("email");
    }else{
        $select_use = $conn->prepare('SELECT * FROM `users` WHERE name = ?');
        $select_use->execute([$userName]);
        /*$row = $select_user->fetch(PDO::FETCH_ASSOC);*/
        $ro = $select_use->fetch();
        if($select_use->rowCount()>0){
            echo json_encode("username");
        }else{
            $insert_user = $conn->prepare("INSERT INTO `users`(name, email,
password) VALUES(?,?,?)");
            $insert_user->execute([$userName, $userEmail, $submitted_pw_hash]);
            echo json_encode("success");

        }

    }
```

In ../js/usersignup.js

Si controlla il contenuto del file json e se la mail o lo username esistono già viene restituito un messaggio all'utente, altrimenti viene indirizzato alla pagina di login

```
dataType: "json",
                success: function(response){
                console.log(response);
                if(response == "success"){
                      console.log("success");
                      location.href = "../html/userlogin.php";
                }else if(response == "username"){
                  $("#response").children().text("Lo username esiste già");
                  $("#response").show().fadeOut(2500);
                }else {
                  $("#response").children().text("L'email è già registrata");
                  $("#response").show().fadeOut(2500);
                }

              },
```