Progetto Tweb

Silvestro Stefano Frisullo
mat. 832813

31 agosto 2019

Indice

1 Tema e sezioni del sito **1**

2 Funzionalità **1**
2.1 Login 1
2.2 Logout 2
2.3 Registrazione 2
2.4 Gestione del contenuto generato dall'utente 2
2.4.1 Wishlist 2
2.4.2 Carrello 3

3 Caratteristiche **3**
3.1 Usabilità 3
3.2 Animazione 4
3.3 Sessioni 4
3.4 Interrogazione del database 4
3.5 Validazione dati input e sicurezza 5

4 Front-end **5**

5 Back-end **6**

1 Tema e sezioni del sito

Il tema del sito è un e-commerce di scarpe. È possibile scegliere tra diversi modelli, classici o sportivi e per uomo o donna. Per ogni modello è presente nome, descrizione, taglia disponibile e prezzo.

Le sezioni *commerciali* del sito sono:

- Man
- Woman
- Classic
- Sport
- Sneakers

La parte superiore presenta 4 pulsanti:

- Home
- Login/Logout
- Wishlist
- Carrello

2 Funzionalità

2.1 Login

Appena l'utente accede ad una qualsiasi pagina del sito, se non ha una sessione valida attiva, viene reindirizzato su *login.html*. Questa pagina prevede due opzioni: il login tramite username e password e la registrazione. L'utente può loggarsi inserendo username e password oppure essere autenticato dal sistema se in possesso del cookie di login persistente. I file coinvolti sono: *login.js* e *login.php*. Nel caso di login tradizionale javascript manda una ajax request al server, il quale farà query sul database, e in caso affermativo autenticherà l'utente, creerà la sessione e il cookie di login persistente valido 24 ore. Il cookie è un hash con sha256 di username + timestamp Unix + un numero casuale tra 0 e 100000000 generato con *random_int*. Nel caso di autenticazione con cookie,*login.js* manderà ajax request col contenuto di tale cookie e se uguale a quello nel database l'utente è accettato.

2.2 Logout

Per fettuare il logout, bisogna cliccare suul'icona a forma di utente. La pagina che gestiste il logout è sempre *login.html*, ma opportunamente modificata dal codice javascript di *login.js*, in modo da informare l'utente di essere già autenticato tramite un'immagine e la scritta *Already logged in ;)* . Al logout viene cancellato il cookie persistente di login, quindi bisognerà reinserire username e password.

2.3 Registrazione

Per la registrazione sono coinvolti i file *login.js* e *register.php*. Javascript manda i campi non vuoti al server che controlla che l'username non sia già occupato e libero crea il profilo utente. Successivamente l'utente può loggarsi.

2.4 Gestione del contenuto generato dall'utente

I contenuti generabili dall'utente sono la **wishlist** e il **carrello**.

2.4.1 Wishlist

La **wishlist**, a cui si accede cliccando sull'icona del cuore, è gestita dai file *wishlist.html*, *wishlist.js*, *wishlist.php*, *add-wishlist.php*, *remove-wishlist.php*. Per scelta progettuale, la wishlist è memorizzata all'interno di una tabella nel database e non è possibile aggiunge lo stesso oggetto più volte. Riprendendo lo stile della home, la wishlist permette di visualizzare le scarpe, aggiungerle o rimuoverle. Javascript manda un'ajax request senza dati a *wishlist.php*, il quale prenderà da $_SESSION l'id dell'utente. Con l'id utente fa query e ricava tutte le scarpe, crea gli oggetti *Shoe* corrispondenti e le restituisce in formato JSON secondo questa sintassi:

```
[{ "id":"3", //id univoco scarpa
   "model":"Classic",
   "price":"200",
   "description":"perfect for classy man",
   "image":"images\/scarpa-classica.jpg" // percorso relativo
      immagine
   }]
```

in caso di successo, javascript rappresenta ogni scarpa in html e inserisce il codice nel corpo della pagina. In *wishlist.html*, ogni scheda ha un bottone *Remove*, che tramite *removeWishlist* manderà una ajax request a *remove-wishlist.php*, con id del prodotto da rimuovere dalla wishlist (quindi ci sono

due ID, uno è dell'utente ed è salvato in $_SESSION, l'altro è della scarpa e si trova in $_GET). *remove-wishlist.php* farà una query sul database e rimuoverà dalla wishlist la scarpa con quell'ID restituendo "ok". A quel punti javascript eliminerà il nodo DOM dell'oggetto e aggiornerà la pagina. Per aggiungere un oggetto alla wishlist, si hanno due opzioni: tascinarlo sull'icona del cuore oppure cliccarci sopra, e in seguito cliccare su "Add to wishlist". I file che gestiscono questo caso sono *wishlist.js* e *add-wishlist.php*. Il primo si occupa dell'animazione e delle chiamate ajax, mentre il secondo tramite id della scarpa fa una query al database. Dato che non è possibile inserire più di una volta lo stesso oggetto, nel caso in cui sià già presente il server restituisce "duplicate", e l'utente è notificato che il prodotto è già in wishlist. Altrimenti, se tutto va bene, javascript rimuove la scheda del prodotto dalla pagina, in modo da dare un feedback visivo della corretta esecuzione dell'operazione, ma senza essere troppo invadenti. Stesso funzionament per quando si aggiunge alla wishlist dalla pagina del prodotto *shop-single.html*, ma in questo caso il file javascript responsabile della ajax request è *single-item.js*.

2.4.2 Carrello

Diversamente dalla *wishlist*, il carrello è implementato tramite variabili di sessione. Gli oggetti possono essere aggiunti o rimossi, e l'aggiunta come per la wishlist può avvenire trascinando gli oggetti o cliccando sul bottone relativo. L'array che contiene i dati del carrello è $_SESSION["items"] e i file coinvolti sono *cart.html*, *cart.js*, *request.js*, *show-single.js*, *cart.php*, *add-cart.php*, *remove-cart.php*. In fondo alla pagina è indicato il costo totale degli oggetti nel carrello e il bottone "Buy!" simula una vendita fittizia, fa comparire un'immagine di conferma all'untente e svuota il carrello. Cliccare sul bottone Buy con carrello vuoto farà apparire la scritta *You need to add something to the cart first!*. Per l'aggiunta al carrello, sia tramite trascinamento che tramite bottone, viene usato l'id del prodotto. Tale id, per scelta progettuale, è inserito come parametro GET nella url. Ciò consente alla funzione getSearchParameters() di recuperarlo e usarlo come parametro nella ajax request allo script *add-cart.php*. Stesso funzionamento, ma con effetto contrario, vale per la rimozione dal carrello.

3 Caratteristiche

3.1 Usabilità

L'utente è informato dell'avvenuto successo o fallimento di ogni operazione legale:

1. Login/Logout
2. Registrazione
3. Aggiunta/Rimozione da carrello e wishlist

Il feedback è dato da un'immagine con messaggio che appare al completamento dell'operazione, tranne nel caso dell'aggiunta a carrello e wishlist con drag & drop, in cui ho preferito far scomparire la scheda l'oggetto appena aggiunto, per dare un senso più realistico di drop e allo stesso tempo un feedback poco invasivo. Le unica operazione in cui il feedback è dato tramite popup di alert è l'aggiunta alla wishlist di un oggetto già presente. L'aggiunta al carrello dello stesso oggetto non crea alcun problema e può essere trasparente all'utente in quanto, trattandosi di un array associativo, una chiave già presente non ne modifica il contenuto.

I colori scelti sono in tonalità pastello per una facile lettura e ho evitato contrasti troppo forti.

3.2 Animazione

Ho implementsto il drag & drop di oggetti nel carrello e nella wishlist usando la libreria JQuery UI. I metodi utilizzati sono droppable e draggable. Il primo permette di definire un oggetto come droppable e associarvi un evento nel caso di drop, che a seconda dei casi, è un'ajax request a *add-cart.php* oppure ad *add-wishlist*. Inoltre permette di specificare quando è rilevato il drop: in questo caso perchè sia rilevato, il cursore deve essere all'interno dell'area dell'oggetto. Draggable invece permette ad un oggetto di poter essere rilasciato , definisce lo zIndex, e posiziona il cursore alla base dell'area dell'oggetto, in modo da far vedere all'utente cosa sta puntando per facilitare il drop.

3.3 Sessioni

Le sessioni sono utilizzate per mantenere il login e gli oggetti nel carrello. Il file che si occupa di controllare che la sessione sia valida è *session.js*, che manda una ajax request a *session.php* e viene chiamato in ogni pagina del sito. Se la sessione non è valida l'utente viene reindirizzato alla pagina di login.

3.4 Interrogazione del database

Il database è interrogato in numerose query, le principali sono:

- **category.php**: $query = "SELECT * FROM ‘shoe‘ WHERE gender ="".$gender."' OR type= "".$type.""";
  Restituisce le scarpe per tipo o per uomo/donna
- **add-wishlist**: $query="INSERT INTO ‘wishlist‘ (‘UserID‘, ‘shoeID‘) VALUES ('$id', '$shoeID');";
  Inserisce nella tabella wishlist un'associazione ID utente - ID scarpa
- **login.php**: "SELECT password FROM user WHERE username='$name'";
  cerca la password di un utente dato username
- **wishlist.php** $query =
  "SELECT DISTINCT shoe.shoeID,shoe.brand, shoe.size, shoe.model,shoe.image, shoe.gender,shoe.type, shoe.description, shoe.price FROM shoe JOIN wishlist ON wishlist.shoeID WHERE shoe.shoeID=wishlist.shoeID AND wishlist.UserID='$id'";

3.5 Validazione dati input e sicurezza

La validazione è fatta lato client controllando che i form non siano vuoti e lato server ripulendo le stringhe con *real_escape_string()* prima di fare la query

Per quanto riguarda la sicurezza la password utente viene salvata in forma di hash sha256, il cookie di login reso più sicuro con salt = data Unix + numero random, le variabili di sessione controllate ad ogni pagina.

4 Front-end

Ho diviso i file in modo molto rigido in base alla tipologia, html, css, js, php, immagini della scarpe e icone. Tutti i file si trovano nella rispettiva cartella. La prima pagina che viene eseguita è *index.html*, la quale caricherà *session.js* per controllare la sessione , poi *includer.js*, che serve per riempire il top, il body ed il footer poi e infine *request.js* che popolerà la home page con i prodotti.

shop-single.html mostra il prodotto selezionato nello specifico, con immagine più grande e pulsanti per aggiunta a carrello o wishlist. *category.html* invece mostra i prodotti in base alla categoria selezionata nel menù. Le dimensioni dei contenitori sono tutte adattive e lo schema a schede si adatta bene anche a schermi di dimensioni ridotte. Lo stile è quanto più possibile unobtrusive, il codice html contiene solo richiami a file javascript, mentre solo

in alcuni punti del codice javascript è inserito del codice html da mettere in pagina.

5 Back-end

Il codice php ha una sola classe , Shoe, che serve a creare degli oggetti shoe da convertire più agilmente in JSON.

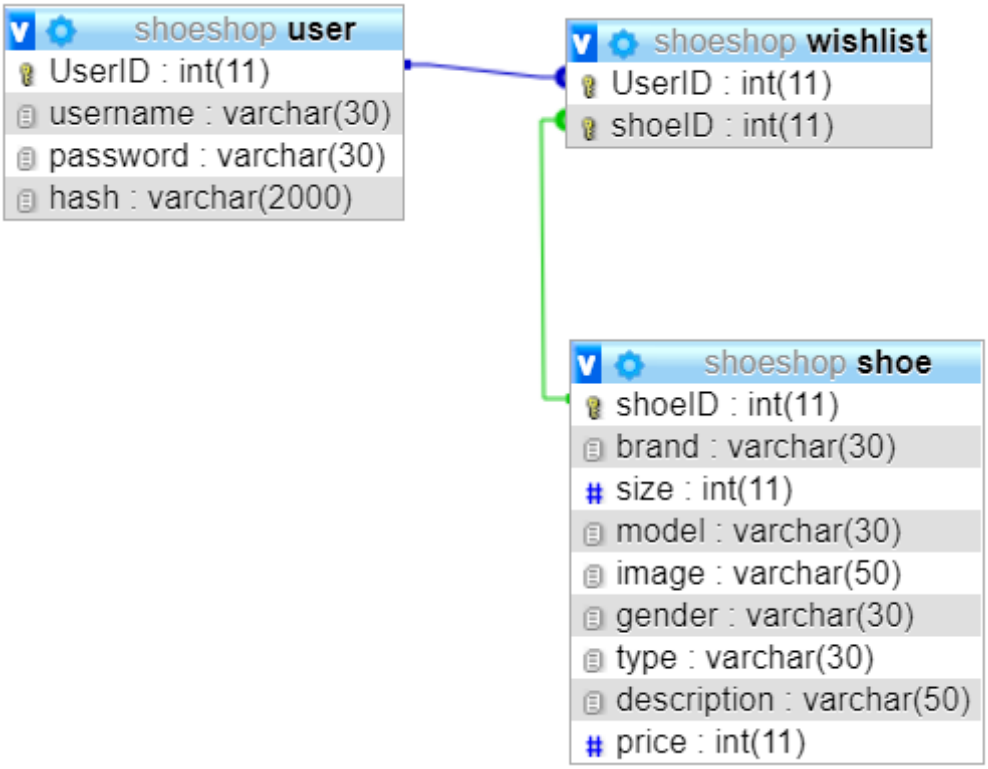

Figura 1: Schema del database

Nome Script	Parametri GET	Descrizione
add-cart	id (scarpa)	aggiunge scarpa al carrello
remove-cart	id (scarpa)	rimuove scarpa da carrello
add-wishlist	id	aggiunge scarpa a id
cart	nessuno	JSON di tutte le scarpe nel carrello
category	type, gender	JSON delle scarpe corrispondenti
checklogin	cookie	logged se loggato NOT altrimenti
login	login / username password	logged se loggato not se input errato
logout	nessuno	termina la sessione e cancella cookie
register	jusernameRegister passwordRegister	Output: ok, username taken,error
remove-cart	id	Rimuove scarpa dato id
remove-wishlist	id	Rimuove scarpa dato id
retrieve	nessuno	ritorna tutte le scarpe in JSON
session	cookie, username	Controlla se la sessione è valida
single-retrieve	id	Output: logged, not restituisce scarpa dato id in JSON
wishlist	nessuno	ritorna JSON delle scarpe in wishlist utente

Le condizioni di errore sono gestite in javascript a seconda dei casi con un feedback all'utente