Cortese Gabriele 858372

Relazione del progetto di Tweb

Aspetti progettuali e tecnici della piattaforma GCollection

di.unito.it

1. Tema del sito:

La piattaforma sviluppata è la base di un e-commerce nell'ambito della vendita di capi d'abbigliamento. Si da la possibilità all'utente di iscriversi o effettuare il login potendo, successivamente, osservare i prodotti a disposizione raggruppati in categorie, ovvero T-shist, Caps, Pants.
Gli obiettivi principali del sito sono il mettere a esposizione prodotti nel modo più minimal possibile, tramite un design pulito e fluido, dare la possibilità di usare un carrello, una wishlist per salvare i prodotto che piacciono di più o destinati all'acquisto e far usare di una sezione dedicata ai commenti per la community.

2. Sezioni principali:

La piattaforma è logicamente suddivisibile in 6 sezioni principali:

1. *Login* —> La prima ad essere visualizzata dagli utenti è la pagina di login all'interno della quale si da la possibilità di effettuare l'inserimento delle proprie credenziali, tramite form di login e signin, visitando successivamente la index.
2. Index —> Tramite la index si possono vedere i prodotti in vetrina. Rappresenta il punto dal quale è possibile muoversi per tutte le altre sezioni quali vedere il dettaglio di un singolo item, controllare il proprio carrello, la wishlist propria e quella degli amici ed infine effettuare il logout.
3. Details —> Facendo click sul singolo prodotto è possibile entrare nella pagina dedicata ad esso. All'interno sono visibili le sue caratteristiche, i bottoni per l'aggiunta al carrello o alla wishlist e una form per l'inserimento di un proprio commento.
4. Shoppingcart —> Cliccando sull'icona, nella parte destra della navbar, è possibile visitare la pagina in cui sono presenti i prodotti del nostro carrello e quelli nella wishlist. In entrambi i casi è possibile manipolarli andando ad eliminare o vedere nel dettaglio gli item in esame.
5. FindWishlist —> La pagina presenta una textbox che permette l'inserimento dell'username dell'utente del quale vogliamo vedere la wishlist. Cliccando sul bottone "Search!" facciamo comparire i prodotti, i quali possono essere spostati, tramite drag&drop, verso l'icona di un carrello che, tramite un bottone di conferma, inserisce il prodotto nel proprio carrello.
6. Logout —> Cliccando sull'icona di logout sulla navabar, parte destra, permettiamo all'utente di uscire dalla piattaforma e di ritornare alla pagina di login, rilasciando cosi tutte le variabili di sessione.

3. Funzionalità:

3.1 *Login/Registrazione*:
Le due operazioni sono state gestite nella stessa pagina tramite la modifica dinamica del DOM. Entrambe le operazioni sono fruibili dalla login.php, la quale al caricamento, mostra la form di login composta da due textbox, un link per fare lo switch alla registrazione ed infine un bottone di conferma credenziali.
Cliccando sulla scritta "signin", il listener dell'evento click() sul link avvia una funzione javascript in grado di modificare gli oggetto della form rendendola adatta ad una registrazione inserendo textbox per username, conferma password e cambiando alcuni attributi dei tag. Il passaggio dalla form di signin a quella di login avviene tramite l'aggiornamento della pagina che porta tutto allo stato originale.
In caso di conferma delle credenziali con il bottone "login/signin" si verra reindirizzati alla index.php.

3.2 *Logout:*
Il logout avviene con click sull'icona presente nella navbar in alto a destra. Non vi è grafica ma semplicemente uno script php in grado di fare unset e destroy delle variabili di sessione, con successivo redirect verso la login.php.

3.3 *Gestione del contenuto*:
Una volta che l'utente è stato correttamente verificato tramite la pagina login.php verrà reindirizzato verso la index.php all'interno della quale potrà osservare i prodotti disponibili. Sulla navbar è presente un menù a tendina dalla quale è possibile gestire la tipologia di prodotti da vedere (T-shirt, Caps, Pants). Il cambiamento dinamico degli oggetti avviene tramite listener e chiamate asincrone tramite gli script presenti nella "effects_index.js".
In questa fase del sito è possibile vedere solamente nome e immagine del prodotto. Ogni item è posto all'interno di un div cliccabile. Al verificarsi di tale evento si passa alla pagina di dettaglio del prodotto,

ovvero la details.php, nella quale sono presenti caratteristiche specifiche come nome, categoria, materiale, colore, taglia e prezzo. L'utente ha la possibilità di creare/organizzare il contenuto in modo personalizzato poichè, nella pagina di dettaglio del singolo item è possibile scrivere/leggere recensioni del prodotto in esame. Nella stessa pagina è possibile, tramite l'uso dei bottoni "add to cart" o "add to wishlist", inserire rispettivamente tale prodotto nel proprio carrello o nella wishlist. Ulteriore personalizzazione è rappresentata dalla pagina "Find wishlist" in cui si da la possibilità di visitare la wishlist di un altro user, scrivendo il suo username, e trasportare fisicamente i suoi prodotti nel proprio carrello. L'utente ha la possibilità di gestire i propri prodotti all'interno della shoppingcart.php in cui è possibile eliminarli, vedere il subtotale e i dettagli dei singoli item.

4. Caratteristiche:

4.1 *Usabilità*:
La presenza della stessa navbar e dello stesso footer permette di essere sempre continui ed evitare quindi di disorientare l'utente. Tutte le pagine hanno un aspetto molto minimal e coerente sia dal punto di vista grafico, poichè sono stati usati pochi colori e contrasti sufficienti per richiamare l'attenzione dell'utente, sia dal punto di vista strutturale poichè tutte le pagine hanno una struttura "a righe" che permette la sistemazione ordinata degli oggetti. Sono state usate anche le caratteristiche di buona progettazione della pagina definite da Krug riguardo alla sua prima legge "don't let me think", andando sviluppare i punti principali quali mettere in rialzo gli oggetti sui quali l'utente sta ponendo l'attenzione, usare stili CSS differenti sugli oggetti cliccabili ed evitare l'affollamento d'informazione. Grazie all'uso del layout "a righe" fornito da bootstrap è stato possibile sfruttare la legge dell'organizzazione figurale nello specifico la legge della vicinanza e della chiusura, questo per far si che gli oggetti della pagina fossero raggruppati in unità per evidenziare un blocco d'informazione precisa (ex: form login/signin, dettaglio.php con l'allineamento foto-dettaglio). Per diminuire il golfo della valutazione, a fronte di ogni interazione con bottoni o link è stata gestita la restituzione di un feedback sia in caso di fallimento che di successo (ex: login/signin, add to cart, add to wishlist, add comment).

4.2 *Animazione*:
La maggior parte degli effetti grafici sono gestiti tramite fogli di stile CSS. Nelle textbox presenti nella login.php sono stati gestiti gli eventi focus e (not)focus, aumentando la dimensione della textbox selezionata colorandone il bordo. Su tutti i bottoni presenti nelle pagine è stato gestito l'effetto hover permettendo il cambiamento di colore di sfondo.
Le animazione gestite tramite javascript sono quelle legate ai feedback, sviliuppati quasi tutti tramite effetti fadeIn e fadeOut. Anche la pagina findWlist.php ne fa uso per far "blinkare" l'immagine del carrello, metodo per attirare l'attenzione e per fare leggere il messaggio posto sopra, in cui si invita l'utente a trasportare gli oggetti sull'immagine. Per lo sviluppo del drag&drop viene usata la libreria jqueryUI, nello specifico tutti i div creati dinamicamente sono dichiarati con attributo class = ui-widget-content per settare loro la capacità di essere trasportabili, mentre l'immagine del carrello definita con class = ui-widget-header. Nel momento in cui l'item d'interesse è portato sul target, ovvero l'immagine del carrello, viene fatto comparire un modal tramite la simulazione di un click di un preciso bottone. Tramite essa è possibile vedere le caratteristiche del prodotto selezionato e confermare o cancellare l'aggiunta nel carrello.

4.3 *Sessioni*:
La variabili di sessione sono create lato server. Le uniche due sono:
$_SESSION["logged"] —> array associativo contenente le informazioni dell'utente appena registrato o loggato. Viene utilizzata in tutte le pagine come metodo di sicurezza, permettendo solo agli utenti già iscritti di visitare la piattaforma, ma anche per alleggerire il carico del database evitando delle query ridondanti.
Contenuto:
- $_SESSION["logged"]["email"]: email dell'utente
- $_SESSION["logged"]["idcarrello"]: id del carrello
- $_SESSION["logged"]["username"]: username
- $_SESSION["logged"]["idwlist"]: id wishlist

$_SESSION["error_login"] —> contiene una stringa d'errore pronta ad essere stampata se contenente qualcosa. Il check del contenuto viene fatto nella pagina login.php e in caso positivo ne viene copiato il contenuto in un tag HTML visibile all'utente. Usata solo per la login.php e nell'index fatta la destroy().

Contenuto:

$_SESSION[“error_login”]: “Database errore occurred, try later.” se la query fallisce per problemi di db.

$_SESSION[“error_login”]: “Username o password not valid.” se la query fallisce per credenziali errate.

4.5 *Interrogazioni database*:

In elenco le pagine php che fanno richieste al database con dettaglio sulle operazioni:

./user

-login_script.php: prese in ingresso email e password, lo script controlla che nel db esista una tupla con tali dati. In caso positivo tutti i dati relativi all’utente vengono messi nella $_SESSION[“logged”] con successivo redirect. In caso contrario, la $_SESSION[“error_login”] viene caricata con una stringa d’errore con redirect verso la login.php.

-logout_script.php: unico scopo è quello di fare unset e destroy di tutte le variabili di sessione senza ricevere o restituire dati.

-newsLetter_script.php: ricevuto in ingresso l’email, ne viene fatta l’insert nella tabella newsletter.

-signin_script.php: riceve in ingresso email, password, conferma password, username. Se esiste già utente con lo stesso username lo script fallisce e ritorna stringa d’errore alla login.php. In caso contrario prosegue con l’inserimento nelle tabelle carrello, wishlist e infine utente.

./products

-getInfoProducts.php: data la categoria dalla pagina index.php, viene ritornata una struttura contenente tutti i dati relativi ai prodotti della categoria scelta.

-getInfoProductFromName.php: dato il nome di un prodotto ne viene ritornata la tupla contenente tutte le sue informazioni.

-getCommentsFromItem.php: fornito nome del prodotto, viene ritornata una struttura contente tutti i commenti legati, con username dell’autore.

-getListItemsWlistFromUser.php: dato username di un utente, viene prima ricercato l’ID della wishlist legata all’username. In caso di successo della query vengono ricercati tutti i prodotti legati a quell’identificativo e ritornati in una struttura dati.

-addCarrello_script.php: forniti i dati di un prodotto, si cercano le stesse informazioni nella tabella Composti, in caso positivo viene fatta update per aumentare di 1 la qta del prodotto ricercato, altrimenti viene inserita una nuova tupla.

-addCommento_script.php: dato il testo e la data del commento appena scritto si inserisce una nuova tupla all’interno della tabella Commenti.

-addWlist_script.php: ricevuti in input i dati di uno specifico prodotto, viene inserito nella wishlist dell’utente.

-deleteItemFromCarrello_script.php: ricevuto in ingresso nome del prodotto e data d’inserimento nel carrello, lo script controlla se esiste tupla con questi dati. Se esistente ne preleva la qta e se > 1 allora fa update decrementandola altrimenti elimina la tupla.

deleteItemFromWlist_script.php : ricevuto nome item e id della wishslist viene elimanta la tupla corrispondente a tali dati. Non viene fatto il controllo sulla qta perché non è presente nella tabella Contiene.

4.6 *Validazione dati in input*:

La maggior parte dei dati in input deriva dalla pagina login.php. In tutte e due le form (login e signin) la validazione viene fatta automaticamente tramite alcuni attributi nei tag, quali required, usato per obbligare l’utente a inserire qualcosa, e l’attributo type che in alcuni casi prende valore “email” per fare il controllo della sintassi coerente con un indirizzo di posta elettronica oppure tipo “password” per nascondere i caratteri.

Altre validazioni sono presenti nel footer di ogni pagina nell’inserimento dell’email per la newsletter.

L’unica textbox poco controllata lato client è quella presente nella pagina findWlist.php in cui l’utente può scrivere qualsiasi cosa, infatti il controllo viene fatto lato server.

4.7 *Sicurezza*:

La sicurezza viene gestita tutta lato server. Ogni dato che viene preso dal lato client viene controllato e prima di essere concatenato ad una query passa attraverso la funzione PDO quote() in grado di eliminare caratteri speciali, questo per evitare l’sql injection. L’eccezione viene fatta per il campo “password” della

login.php, il quale non passa attraverso la quote() ma attraverso la funzione md5(), in grado di cifrare la stringa di testo. Così facendo anche i possibili caratteri speciali verrebbero criptati in lettere e numeri.

5. Front-end:

5.1 *Organizzazione file e cartelle:*

```
├── css
│   ├── stile.css
│   ├── stileDetails.css
│   ├── stileLogin.css
│   ├── stileSearchWl.css
│   └── stileShop.css
├── img ..
├── js ..
│   ├── effects_details.js
│   ├── effects_findWlist.js
│   ├── effects_index.js
│   ├── effects_login.js
│   └── effects_shopcart.js
├── products
│   ├── addCarrello_script.php
│   ├── addComment_script.php
│   ├── addWlist_script.php
│   ├── deleteItemFromCarrello_script.php
│   ├── deleteItemFromWlist_script.php
│   ├── getCommentsFromItem.php
│   ├── getInfoProductFromName.php
│   ├── getInfoProducts.php
│   └── getListItemsWlistFromUser.php
├── shoppingcart.php
├── login.php
├── navbar.html
├── index.php
├── details.php
├── findWlist.php
├── footer.html
└── user
    ├── login_script.php
    ├── logout_script.php
    ├── newsLetter_script.php
    └── signin_script.php
```

5.2 *Separazione presentazione/contenuto/comportamento:*
Nelle pagine HTML non è mai presente del codice javascript, viene separato e messo nella cartella JS come indicato nello schema ad albero nella sezione sopra. Ogni pagina php ha un proprio file JS.

*1)**effects_login.js**:* viene catturato l'evento click solo dei link che permettono lo switch tra form login e signin. Nello specifico, per passare da login a signin, allo scatenarsi del click viene avviata la createFormSignin() in grado di accedere ai nodi della form e modificarne il DOM tramite replaceChild, insertBefore.
Per passare dalla signin alla login da JS non viene gestito nulla perché viene fatto un refrash della pagina portando tutto allo stato originale. Anche gli eventi dei bottoni non sono gestiti tramite javascript ma solo con l'attributo action e metodo della form in cui si trovano.
2)**effects_index.js**: al "ready" della pagina, viene fatta una chiamata asincrona verso getInfoProducts.php per la ricezione dei dati dei prodotti in base alla categoria scelta. In caso di successo la funzione createProductsDOM() agisce sul DOM appendendo i div dinamici. Successivamente la funzione productsDiv() permette di inserire gli effetti shadow per la gestione dei feedback su ogni prodotto.
Viene catturato anche l'evento click() sul bottone submit riguardo la newsletter. Allo scatenarsi, il contenuto della textbox viene fatto passare nella validateEmail() che controlla la sintassi con un'espressione regole e in caso di responso positivo viene fatta una chiamata asincrona verso la newsLetter_script.php.
3)**effects_details.js**: al "ready" viene fatta una richiesta asincrona per reperire tutte le informazioni del prodotto in dettaglio. Solo dopo la ricezione viene avviata la createProductDetailsDOM() che crea dinamicamente la pagina html. In successione viene poi fatta una chiamata AJAX per reperire tutti i commenti legati al prodotto. In caso di successo viene avviata la createCommentsDOM().
Sempre tramite richieste asincrone sono stati gestiti i click dei bottoni di aggiunta in carrello, in wishlist e di un commento, i quali listener avviano rispettivamente sendToCarrello(), sendToWlist(), sendToComment() interagendo con il db sempre in modo asincrono tramite AJAX. In questi tre casistiche, in caso di successo, viene sempre restituito un feedback con effetto fadeIn e fadeOut.
4)**effects_shopcart.js**: l'unica cosa gestita da javascript è la grafica che, in base al parametro presente nell'URL, decide se sottolineare la voce "To order" o "Manage wishlist". La restante parte della comunicazione è fatta tramite l'uso di tag <a> con attributo href.
5)**effects_findWlist.js**: al click del bottone "search" vengono fatte chiamate asincrone per il reperimento degli item presenti nella wishlist. In caso di risposta positiva, per ogni item ritrovato viene eseguita la getInfoProd() che ritorna tutte le sue informazioni. La createProductsDOM() crea e appende dinamicamente i div contenente il prodotto. Di tutti i dettagli dell'item prelevati da db, solo l'immagina e il nome saranno

visibili all'utente, tutte le altre sono nascoste. Verranno mostrate solo in fase di drag&drop degli oggetti sul carrello.
Dopo la creazione dinamica di tutti i div, viene settata, tramite il selettore di classe, la proprietà draggble così da renderli trasportabili. A pagina completa viene resa visibile l'immagine del carrello, dichiarato droppable per essere identificato come obiettivo per gli oggetti draggable. Se il div trasportato accede all'area del carrello compare un modal di riepilogo e conferma del trasferimento del prodotto nel proprio carrello.

6. Back-end e comunicazione front/back-end:

6.1 *Architettura generale classi/funzioni php:*
Non sono presenti classi php ma solo funzioni. Lo schema ad albero rende esplicita la struttura della cartelle.

```
├── css
│   ├── stile.css
│   ├── stileDetails.css
│   ├── stileLogin.css
│   ├── stileSearchWl.css
│   └── stileShop.css
├── img ..
├── js ..
│   ├── effects_details.js
│   ├── effects_findWlist.js
│   ├── effects_index.js
│   ├── effects_login.js
│   └── effects_shopcart.js
├── products
│   ├── addCarrello_script.php
│   ├── addComment_script.php
│   ├── addWlist_script.php
│   ├── deleteItemFromCarrello_script.php
│   ├── deleteItemFromWlist_script.php
│   ├── getCommentsFromItem.php
│   ├── getInfoProductFromName.php
│   ├── getInfoProducts.php
│   └── getListItemsWlistFromUser.php
├── shoppingcart.php
├── login.php
├── navbar.html
├── index.php
├── details.php
├── findWlist.php
├── footer.html
└── user
    ├── login_script.php
    ├── logout_script.php
    ├── newsLetter_script.php
    └── signin_script.php
```

6.2 Descrizione delle funzioni remote:
La descrizione delle funzioni sono scritte nella sezione 4.5 della relazione. Di seguito verranno analizzate input, output, gestione errore e funzioni di callback lato javascript.

./user
-login_script.php
Sintassi input: $_POST["email"], $_POST["passwd"]
Sintassi output: none
Condizioni d'errore:
1)Malfunzionamento DB: $_SESSION["error_login"] = "Database error occurred, try later!"
2)Credenziali errate: $_SESSION["error_login"] = "Username or password not valid"
Funzioni callback JS/AJAX: none, la chiamata viene fatta dalla login.php
Richiamata da: login.php

-newsLetter_script.php
Sintassi input: $_POST["email"]
Sintassi output: none
Condizioni d'errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX:
successo: feedback positivo con stampa a video
errore: feedback negativo con stampa a video "Something went wrong, try later"
Richiamata da: effects_index.js —> sendEmailNl()

./products

-getInfoProducts.php
Sintassi input: $_POST[“nomeCat”]
Sintassi output:

```
[
  {
    "nome":"Aruba",
    "0":"Aruba",
    "idCat":"T-shirt",
    "1":"T-shirt",
    "pathImg":".\/img\/T-shirt0.jpg",
    "2":".\/img\/T-shirt0.jpg",
    "descrizione":"Maglietta interessante",
    "3":"Maglietta interessante",
    "colore":"Nero",
    "4":"Nero",
    "materiale":"Sea Island Cotton",
    "5":"Sea Island Cotton",
    "prezzo":"180",
    "6":"180"
  },
  {…},
  {…},
  {…},
  {…}
]
```


Condizioni d’errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX:
successo: avviate le createProductsDOM() e productsDiv()
errore: feedback negativo con stampa a video “Database error occurred, try later”
Richiamata da: effects_index.js —> document.ready()

-getInfoProductFromName.php
Sintassi input: $_POST[“nomeItem”]
Sintassi output:

```
[
  {
    "idCommento":"35",
    "0":"35",
    "testo":"Molto pesante",
    "1":"Molto pesante",
    "username":"Gabri",
    "2":"Gabri",
    "idProdotto":"Belize",
    "3":"Belize",
    "data":"2020:01:27 10:14:42pm",
    "4":"2020:01:27 10:14:42pm"
  }
]
```

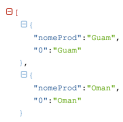

Condizioni d’errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX:
successo: avviata la createProductDetailsDOM()
errore: feedback negativo con stampa a video “Database error occurred, try later”
Richiamata da: effects_details.js —> requestDetails()

-getCommentsFromItem.php
Sintassi input: $_POST[“nomeItem”]
Sintassi output:

```
[
  {
    "idCommento":"35",
    "0":"35",
    "testo":"Molto pesante",
    "1":"Molto pesante",
    "username":"Gabri",
    "2":"Gabri",
    "idProdotto":"Belize",
    "3":"Belize",
    "data":"2020:01:27 10:14:42pm",
    "4":"2020:01:27 10:14:42pm"
  }
]
```

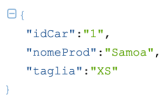

Condizioni d’errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX:
successo: avviata la createCommentsDOM()
errore: feedback negativo con stampa a video “Database error occurred, try later”
Richiamata da: effects_details.js —> requestComments()

-getListItemsWlistFromUser.php
Sintassi input: $_POST[“usrName”]
Sintassi output:



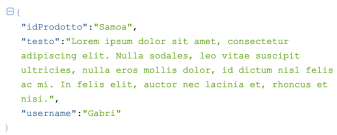
Condizioni d’errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX:
successo: se ritorna vuoto allora viene scritto a video “any products in wishlist”, altrimenti per ogni prodotto trovato viene richiamata la getInfoProd() e successivamente la createProductsDOM().
errore: feedback negativo con stampa a video “An error occurred, check the username”
Richiamata da: effects_findWlist.js —> document.ready()

-addCarrello_script.php
Sintassi input:



Sintassi output: none
Condizioni d’errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX:
successo: feedback positivo stampato a video “add item correctly in shopping cart!”
errore: feedback negativo con stampa a video “Something went wrong, try later!”
Richiamata da: effects_details.js —> sendToCarrello()

-addCommento_script.php
Sintassi input:



Sintassi output:
Condizioni d’errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX:
successo: feedback positivo stampato a video “The comment has been sent!”
errore: feedback negativo con stampa a video “Something went wrong, try later!”
Richiamata da: effects_details.js —> sendToComment()

-addWlist_script.php
Sintassi input:

Sintassi output: none
Condizioni d'errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX:
successo: feedback positivo stampato a video "Add item correctly in wishlist!"
errore: feedback negativo con stampa a video "Something went wrong, try later!"
Richiamata da: effects_details.js —> sendToWlist()

-deleteItemFromCarrello_script.php
Sintassi input: $_GET["nome"], $_GET["date"]
Sintassi output: none
Condizioni d'errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX: none, la chiamata viene fatta dalla shoppingcart.php
Richiamata da: shoppingcart.php

-deleteItemFromWlist_script.php
Sintassi input: $_GET["nome"],
Sintassi output: none
Condizioni d'errore:
1)Malfunzionamento DB: Gestione lato javascript
Funzioni callback JS/AJAX: none, la chiamata viene fatta dalla shoppingcart.php
Richiamata da: shoppingcart.php

6.3 Schema relazionale database

Compost(**idCarrello,nomeProf,dataOrd**,qta,taglia)
Commenti(**idCommento**, testo,username,idProdotto,data)
Ordine(**idOrdine**,data,idProdotto,username,qta,taglia)
Contiene(**idWlist**,nomeProd)
Categoria(**id**)
Prodotto(**nome**, idCat, pathImg, descrizione, colore,materiale)
Utente(email, **username**,pwd,idcarrello,idwlist)
Carrello(**id**)
Wishlist(**id**)
NewsLetter(**id**)