Relazione del progetto di TWEB 2021/2022

Xiao Zhao matr. 913828

Il sito realizzato, chiamato Detra, è un sito simile al subito.it, che si occupa di vendere degli oggetti in seconda mano.

Gli utenti registrati possono visualizzare tutte le offerte che sono presenti nel database, e possono anche pubblicare la loro offerta personale. Volendo possono anche aggiungere le offerte nei loro carrelli come preferiti.

Sezioni principali:

- Home: la pagina principale, all'inizio verrà visualizzata tutte le offerte, si potrebbe anche cercare le offerte per una parola chiave o per categoria.
- My offer: in questa pagina verranno visualizzate solo le offerte pubblicate dall'utento loggato.
- Cart: in questa pagina verranno visualizzate le offerte che sono state aggiunte nel carrello.
- Publish new offer: in questa pagina si può aggiungere una nuova offerta, inserendo titolo, prezzo, categoria, descrizione e immagine.
- Offer: non è una pagina tra i menu, ma si raggiunge cliccando su una offerta qualunque. In questa pagina, verranno visualizzate le informazioni più dettagliate rispetto al home; si può anche procedere per cancellare una offerta o aggiungerla al carrello.

Funzionalità:

- **Login:** viene effettuato nella pagina login.php e gestito con login.js. Inserendo e validando i dati, verranno trasmessi al loginCheck.php con una richiesta di tipo POST. Dopo che il loginCheck.php ha ricevuto le informazione, verrà fatto una richiesta al database per controllare le credenziali. Se è corretto allora verrà inizializzato una sessione, e all'utento verrà dato accesso per entrare nel sito. Nel caso contrario, verrà segnalato il tipo di errore.
- **Logout:** viene effettuato cliccando l'opzione logout nel menu. È presente in tutte le pagine e gestito in un file utils.js comune. Verrà fatto una richiesta di GET al file logout.php che chiuderà la sessione attuale. Alla fine l'utente verrà reindirizzato alla pagina di login con un messaggio di avvenuta logout.
- **Registrazione:** viene effettuato nella pagina register.php e gestito da register.js. Si può anche accedere dalla pagina di login selezionando apposita opzione. Inserendo e validando i dati, verranno trasmessi al addUser.php con POST. In quest'ultimo viene aggiunto un nuovo utente nel database. Verrà visualizzato un messaggio di feedback e reindirizzato alla pagina login sia nel caso di successo sia di fallimento per motivo di account esistente.
- **Gestione del contenuto generato dall'utente:** Un utente può sia aggiungere un'offerta nel carrello personale e sia pubblicare una nuova offerta. La pubblicazione viene effettuata nella pagina publish.php e gestita da publish.js, e attraverso POST, manderà le informazioni al publishOffer.php. Il quale inserirà una nuova offerta nel database che potrà essere visto da tutti gli utenti. Ci sarà un messaggio di feedback per questa operazione.

Caratteristiche:

- **Usabilità:** il menu principale è posizionato in alto a destra, passando con il mouse verranno visualizzate tutte le opzione. Questo è per minimizzare la possibilità di distrarre e di concentrarsi sul contenuto principale. Ad ogni azione che non si ha un feedback diretto, verrà fornito un messaggio breve di feedback. Viene solo usato una finestra di pop-up nel caso di cancellazione dell'offerta, per evitare i clicchi occasionali. Intero sito usa solo l'azzurro come colore principale, e rosso solo per informazioni importanti.
- **Interazione/animazione:** nelle pagine home, myOffer e cart, ho aggiunto un'animazione agli immagini delle offerte usando il metodo animate() di JQuery. Passando il mouse su queste immagini, verranno ingranditi per dare un anteprima più dettagliato dell'articolo.
- **Sessioni:** la sessione è gestita nel lato server di PHP mediante metodi di session e $_SESSION. Viene creata una sessione ogni volta che viene effettuato un login con successo. Tale sessione verrà distrutta quando viene effettuato logout.
- **Interrogazione nel DB:** viene effettuato in quasi tutte le pagine, per visualizzare una lista di offerte, per pubblicarne una nuova e per cancellare. Le query vere e proprie si trovano nei vari file .php con diverse funzioni. Nella pagina home.php si può inserire dei dati per effettuare una ricerca nel DB.
- **Validazione dei dati di input:** tutti gli username sono di tipo email e il prezzo può essere solo di tipo intero. Questi sono validati tramite espressione regolare.
- **Sicurezza:** nel server viene usato la funzione quote() sulle variabili prima effettuare operazioni sul DB. Tutte le informazione pubblicate sulla pagina che sono derivate da un input dell'utente, sono inserite con la funzione text() e non html(), per non alterare la struttura della pagina.
- **Presentazione:** il sito è progettato in maniera intuitiva. Tutti i comandi sono posizionati nella parte alta della pagina, e il contenuto in mezzo, lasciando sempre spazi vuoti tra i componenti. Quasi tutte le dimensioni sono impostate con il %, adottando un layout flessibile.

Front-end:

- **Separazione :** il contenuto è gestito dai file html e php, i quali dichiarano id e class degli elementi ed è privo di attributi come style o degli script. La presentazione è gestita esclusivamente dai file css. Il comportamento è gestito dai JS e JQuery.
- **Cross-platform:** se il sito viene visualizzato su una finestra piccola per simulare una finestra di cellulare, verrà cambiato il file di css e nascosto degli informazioni secondari. Ma la struttura è sempre la stessa per mantenere la famigliarità.
- **Organizzazione:** le cartelle sono suddivise per funzionalità:
    - ./css per presentazione, contiene tutti file css
    - ./html per il contenuto, contiene tutti file html e alcuni di php che servono per includere top, banner, bottom.
    - ./img contiene tutte le immagini necessarie al sito web
    - ./js per il comportamento, contiene tutti i file js
    - ./php per il server, contiene tutti i file che gestiscono le richieste ajax dai file js

- ./sql contine il file database

Back-end e comunicazione front/back-end:

- **Architettura generale classi/funzioni php:** le funzioni php sono suddivise per funzione in vari file php dentro la cartella ./php. Nel file utils.php contiene 2 funzioni comuni, uno per la connessione al DB e l'altro per controllare se l'utente è loggato. Tutti gli altri file sono nominato in modo auto esplicativi.
- **Schema del DB:**

  Roles(id, name)
  Offerts(id, title, category, description, price, image*, user_id)
  Users(id, email, password, role_id)
  Carts(id, user_id, offer_id)

  Con vincolo di integrità referenziale fra:
  L'attributo «user_id» in offerts e la chiave di users.
  L'attributo «role_id» in users e la chiave di roles.
  L'attributo «user_id» in carts e la chiave di users.
  L'attributo «offer_id» in carts e la chiave di offerts.

- **Descrizione delle funzioni remote:**
  - **addUser/addUser():** aggiunge un utente nuovo quando viene effettuato la registrazione con il metodo POST su html/register.php.
    Parametri: username e password.
    Verrà prodotto un messaggio di feedback per il risultato dell'esecuzione.
  - **deleteOffer/deleteOffer():** cancella una certa offerta prendendo id dell'offerta con il metodo GET da url (http://localhost/Progetto_TWEB_21_22/html/offer.php?id=2).
    Parametro: id.
  - **getOffers/printShowOfferToJSON():** preleva le informazioni dettagliate di un'offerta con il metodo POST su html/offer.php
    Parametro: id.
    Output è di tipo json, indicando anche il ruolo dell'utente attuale e se l'articolo è già nel carrello:

```
{
   "admin": "se è admin",
   "cartCheck": "se è nel carrello",
   "offers": [
      {
         "title": "titolo dell'offerta",
         "time": "data dell'offerta",
         "description": "descrizione dell'offerta",
         "price": "prezzo dell'offerta",
```

```
            "image": "percorso dell'immagine dell'offerta",
            "email": "email del venditore"
        }
    ]
}
```


- **getOffers/printOfferToJSON():** preleva le informazioni di tutte le offerte che soddisfano un certo criterio inserito con il metodo POST su html/home.php.
  Parametro: search (la parola chiave), category(categoria dell'articolo).
  Output è di tipo json, un array di offerte. Ha una struttura simile a quello precedente.
- **getOffers/printMyOfferToJSON():** preleva le informazioni sulle offerte dell'utente attuale con il metodo POST su html/myOffers.php.
  Parametro: name (nome utente salvato nella sessione).
  Output è di tipo json, un array di offerte. Ha una struttura simile a quello precedente.
- **getSetCart/printCartToJSON():** preleva le informazioni sulle offerte salvate nel carrello dell'utente attuale con il metodo POST su html/cart.php.
  Parametro: name (nome utente salvato nella sessione).
  Output è di tipo json, un array di offerte. Ha una struttura simile a quello precedente.
- **getSetCart/setCart():** aggiunge un'offerta nel carrello dell'utente attuale con il metodo POST su html/offer.php.
  Parametro: name (nome utente salvato nella sessione), id (id dell'offerta).
- **getSetCart/deleteCart():** cancella un'offerta dal carrello dell'utente attuale con il metodo POST su html/offer.php.
  Parametro: name (nome utente salvato nella sessione), id (id dell'offerta).
- **loginCheck/pwdVerify():** controlla la veridicità dei credenziali inseriti con il metodo POST su html/login.php.
  Parametro: username, pwd.
- **publishOffer/publish():** aggiunge una nuova offerta nel database utilizzando i dati inseriti dall'utente, con il metodo POST su html/publish.php.
  Parametro: title, category, description, price, image, username(dalla sassione).
  Output è un messaggio di feedback.
- **utils/isLogged():** controlla se nella sessione esiste già un utente loggato.
- **Utils/dbconnect():** istanzia una connessione al database creando un nuovo PDO.