Pasta&Basta

Relazione progetto Tweb 2022/2023

Magrì Daniela - 945982

1. Tema del sito e sezioni principali

1.1. Tema del sito: il sito permette la vendita di piatti tipici di pasta di ogni regione d'Italia.
L'utente sceglie e acquista da un menu' il pasto che preferisce, pagando alla consegna da parte di un corriere.
Gli obiettivi che il sito si pone sono quelli di fornire all'utente strumenti per visualizzare ed acquistare i prodotti proposti, come ad esempio il carrello, gestire i propri ordini e tenere traccia degli acquisti effettuati.

1.2. Sezioni principali: il sito è strutturato in 5 pagine, la pagina principale (index.php), la pagina di registrazione (formRegister.php), la pagina dove l'utente visualizza i suoi dati (formMyAccount.php), la pagina del carrello (formCart.php) e infine la pagina di login (formLogin.php).

- *index.php: Navigazione principale,* nella pagina principale troviamo la navigazione principale del sito, tramite le voci presenti in essa è possibile accedere alle 7 sezioni principali dell'homepage. Le voci sono le seguenti:
  - *Home:* in questa sezione l'utente può visualizzare una parte dei prodotti in vendita, in particolare quelli in offerta, descritti dal nome, dagli ingredienti del piatto, dalla regione di provenienza, dal prezzo non scontato e da quello scontato;
  - *Piatti proposti:* in questa sezione l'utente può visualizzare tutti i prodotti presenti nel menu', descritti nello stesso modo indicato nella home;
  - *Orario:* in questa sezione l'utente può visualizzare gli orari di apertura e chiusura del locale, ovvero gli orari in cui è possibile acquistare;
  - *Chi sono:* in questa sezione l'utente può visualizzare informazioni sul venditore/chef di Pasta&Basta, tramite una breve descrizione di esso;
  - *Contatti:* in questa sezione l'utente può visualizzare una serie di recapiti;
  - *La mia etica:* in questa sezione l'utente può visualizzare informazioni riguardo la sostenibilità e i valori su cui si basa tale progetto di ristorazione;
  - *My Account:* in questa sezione, visibile solo agli utenti non admin, l'utente può chiedere di essere indirizzato ad una pagina dove visualizza i dati riguardanti il suo account.

- *formRegister.php: registrazione utente,* attraverso questa pagina l'utente può registrarsi al sito compilando i campi di un form.
- *formLogin.php: accesso utente*, attraverso questa pagina l'utente può accedere al sito, specificando email e password.
- *formMyAccount.php: dati utente,* la pagina dedicata all'utente è suddivisa in 2 sezioni:
    - *I tuoi dati:* in questa sezione l'utente può visualizzare i dati relativi al suo account inseriti al momento della registrazione, in particolare il nome, il cognome, l'email ed il numero di telefono;
    - *I tuoi ordini:* in questa sezione l'utente può visualizzare le informazioni riguardo gli acquisti da lui effettuati;
- *formCart.php: carrello utente,* attraverso questa pagina l'utente può visualizzare, modificare, gestire ed inviare un ordine.

2. Funzionalità

Le richieste HTTP vengono fatte tutte tramite ajax in modo asincrono, utilizzando JSON ed il metodo POST.
Inoltre tutte le query che modificano il database vengono effettuate prevenendo attacchi di SQL_INJECTION e HTML_INJECTION sfruttando la funzione bindParam dell'oggetto PDOstatement.

2.1. Login: l'operazione di login è strutturata nella pagina formLogin.php ed è eseguita nel file login.php.

- *Struttura:* viene usato un fieldset dove al suo interno troviamo un form per inserire l'email, uno per la password ed un pulsante "accedi" per effettuare l'accesso. Tutti i campi sono required e gli eventuali errori di validazione vengono visualizzati mostrando un testo, il cui contenuto cambia in base al tipo di errore, evidenziato in rosso.

La richiesta HTTP invia i dati al server, il file login.php riceverà quindi i dati ed eseguirà l'operazione.
La validazione dei campi per accedere viene fatta sia lato client che lato server.
Lato client, nel file loginU.js, si controllano i valori inseriti nel form per l'email e per la password, tramite una espressione regolare, nel caso in cui quest'ultima non venisse rispettata viene generato un errore di validazione.
Lato server, si verifica che l'utente abbia inserito email e password corrette. Inoltre, tali dati, vengono controllati dal server tramite la funzione strip_tags(), al fine di evitare attacchi XSS.
Al login effettuato con successo l'utente viene ridirezionato alla pagina principale.

2.2. Logout: l'operazione di logout è implementata solo lato server, senza una struttura di appoggio. Consiste in pochi semplici passaggi:

- Cancello tutte le variabili della sessione corrente con "session_unset()";
- Distruggo la sessione corrente con session_destroy();

- Ridireziono l'utente alla pagina principale.

Al fine di ridurre la prevedibilità del "session id" quando creo una nuova sessione, dopo aver distrutto la precedente, inserisco la chiamata alla funzione session_regenerate_id(TRUE);

2.3. Registrazione: l'operazione di registrazione è strutturata nella pagina formRegister.php ed è eseguita nel file register.php. Concettualmente è strutturato come formLogin.php, ovvero viene usato un fieldset dove al suo interno sono presenti più form, anch'essi tutti required, per ogni dato richiesto per registrarsi: il nome, il cognome, l'email, la password ed il numero di telefono, e al fondo un pulsante "registrati" per effettuare la registrazione.
La validazione dei campi viene effettuata prima lato client e solo se questa ha successo verranno inviati i dati al server, che farà la sua validazione.
Lato client, nel file registerU.js, si controlla che ogni campo sia valido, anche qua, tramite una espressione regolare, nel caso in cui tutti i dati siano validi allora viene inviata una richiesta HTTP al server, il file register.php riceverà i dati ed eseguirà l'operazione; altrimenti verrà generato e visualizzato sotto al form incriminato un errore di validazione.
Lato server, si verifica che l'utente non esiste e viene utilizzata la funzione strip_tags().
Nel caso di registrazione avvenuta con successo l'utente viene ridirezionato alla pagina di login.

2.4. Gestione del contenuto generato dall'utente: index.php, formMyAccount.php, formCart.php, ogni utente può navigare asincronicamente, grazie alle chiamate ajax tra le sezioni.
Viene usata la funzione load() di jQuery, che permette di caricare in modo dinamico il contenuto della sezione richiesta, senza dover aggiornare tutta la pagina. Distinguiamo il contenuto da caricare attraverso uno switch-case(idContenuto), in base al caso in cui incombiamo viene inviata una richiesta al file "nomeRichiesta.php" che la eseguirà, ad esempio l'utente schiaccia nella barra di navigazione "Piatti proposti", allora verrà inviata una richiesta al file "products.php".
Le richieste di modifica e conferma di modifica del prodotto, praticabili come admin, vengono inviate ad un unico file ("modifyP.php") specificando, lato client, un flag che identifica l'operazione che il server andrà poi ad eseguire, il tutto con metodo POST.
Le operazioni del carrello vengono gestite allo stesso modo, quindi tra i dati passati al file "cartOp.php", ci sarà anche un flag identificativo.
Il server sfrutta le informazioni contenute nell'array "SESSION", inizializzato al momento del login; le query e le risposte infatti, vengono personalizzate in relazione all'accesso dell'utente o al suo ruolo.
La risposta del server è creata in formato array e trasformata poi in oggetto json, ed è così strutturata: uno status, che può contenere valore 1 (successo) o valore -1 (fallimento) ed altri contenuti in base alla richiesta, ad esempio una "answer", il risultato della query, i dati dell'utente ecc.
Il client successivamente, laddove serve, costruisce e appende dinamicamente il contenuto da caricare, un elemento HTML inserendo così i dati della risposta ottenuta ed eventuali bottoni, con i relativi eventi associati al click di essi.

3. Caratteristiche

3.1. Usabilità: ad ogni operazione il sito fornisce sempre un feedback, sia in caso di errore che in caso di successo. Inoltre ogni volta che viene richiesta una pagina o una sezione, nel tempo di caricamento viene mostrato uno spinner fino a che il contenuto non viene caricato.
Il testo, i pulsanti, le immagini, ogni componente del sito è studiata graficamente per rendere il giusto contrasto tra di loro, in modo tale da poter essere ben distinte.
Per la navigazione viene usata una navbar, la quale indica la posizione dell'utente durante il suo utilizzo evidenziando la sezione in cui si trova.

3.2. Interazione/Animazione: quando l'utente che ha fatto accesso preme il pulsante "aggiungi al carrello", se l'operazione ha successo, il pulsante premuto scompare e viene al suo posto eseguita un'animazione: appare l'immagine rappresentante il prodotto, che si dissolverà linearmente, insieme all'icona di un carrello vuoto, al termine dell'animazione quest'ultima verrà sostituita dall' icona di un carrello pieno, per simulare visivamente l'operazione di inserimento appena svolta.
L'animazione è creata tramite la funzione animate() di JQuery.

3.3. Sessioni: le sessioni utente sono implementate tramite il comando "if(!isset($_SESSION)){ session_start() }" inserito all'inizio di ogni file in cui utilizzo variabili di sessione, nello specifico nei file: connect.php, top.php, navigation.php, formLogin.php, formMyAccount.php, formRegister.php e formCart.php.
In questo modo verifico se una sessione è stata avviata, nel caso non lo fosse riprendo quella corrente, con session_start(), o ne avvio una nuova.
Quando effettuo il login il server estrapola dal database, con una query SQL, i seguenti dati dell'utente: id, username (nome e cognome), email, isAdmin e numero di telefono, che vengono poi memorizzati nell'array di sessione.
Quando effettuo il logout, cancello tutte le variabili di sessione inizializzate precedentemente al momento del login e distruggo la sessione corrente.

3.4. Interrogazioni del database:
Le interrogazioni al database vengono personalizzate e svolte secondo il ruolo che assume l'utente. Prendiamo in esempio le operazioni che si eseguono sui prodotti:

- *Admin:* all'interno della griglia di ogni singolo piatto proposto viene inserito il pulsante "modifica prodotto", al click si visualizzano e si può operare sui dati del prodotto selezionato;
- *Utente che ha fatto accesso (cliente):* all'interno della griglia di ogni singolo piatto proposto viene inserito il pulsante "aggiungi al carrello";
- *Utente che non ha fatto accesso (utente ospite):* all'interno della griglia di ogni singolo piatto proposto viene inserito un alert con su scritto "Per acquistare prima accedi".

In questo modo l'interrogazione al database è ad alto livello e personalizzata.

3.5. Sicurezza: oltre quanto già scritto in precedenza riguardo alla protezione da attacchi XSS, SQL_INJECTION, HTML_INJECTION e la riduzione di prevedibilità del "session id", all'interno del database la password è criptata utilizzando lato server la funzione md5().

3.6. Presentazione: Pasta&Basta dispone delle classiche funzionalità di un sito e-commerce e sono facili ed intuitive da utilizzare.
La presentazione del contenuto è organizzata in un albero di contenitori, creando uno schema a più righe e a più colonne, grazie all'attributo css "display"; vengono inoltre usate percentuali o rem/em per le dimensioni delle componenti, è stato così possibile realizzare un sito responsive sfruttando le media query css.

4. Front-end

Il sito è stato sviluppato dividendo: *presentazione, contenuto e comportamento*.
Il codice javascript viene usato in modo unobtrusive, ovvero è separato da quello HTML, stessa cosa vale per il codice css e jQuery.
Il codice è quindi organizzato per cartelle:

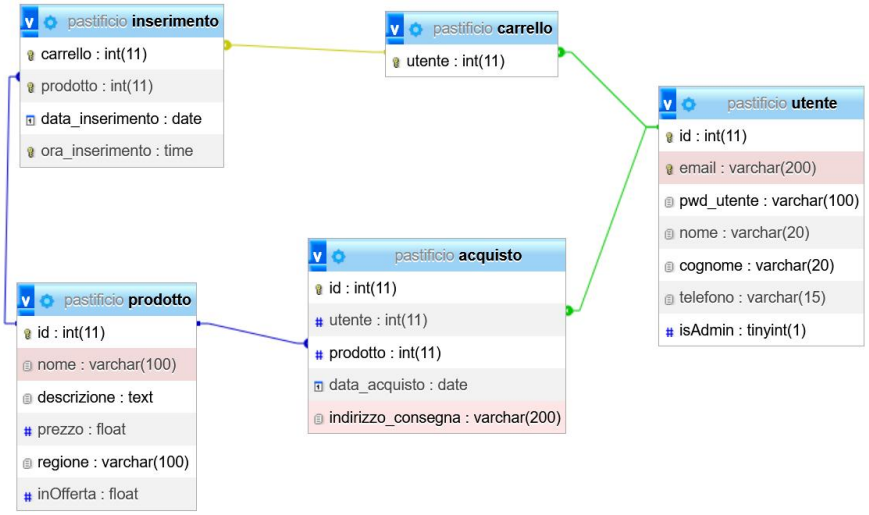
- "files.php" sono contenuti nella cartella "PHP", che a sua volta è suddivisa in sottocartelle (Admin, Database, Users, HomePage) in relazione alla funzione che svolge il file.
- "files.html" sono contenuti nella cartella "HTML", che anch'essa contiene ulteriori sottocartelle (myAccountPage) in relazione alla pagina a cui appartiene lo "scheletro" all'interno del file.

Infine anche i file ".css" e ".js" hanno la loro cartella e vengono importati esternamente all'interno del tag <head> della pagina.

5. Back-end

Ogni operazione che richiede di comunicare con il db, viene gestita da file php diversi, ad eccezione per le operazioni del carrello e dell'Admin, che sono raggruppate in due file: "cartOp.php" e "modifyP.php"; quelle che invece non richiedono di comunicare con il db, sono gestite dal file "response.php" che invia solo una risposta al client.
Ogni file, a parte l'ultimo citato, include il file "connect.php" per la connessione con il database.
"cartOp.php" e "modifyP.php": si occupano della gestione delle richieste ajax generate lato client utilizzando un approccio web-service, ogni richiesta ha un flag e grazie ad esso il server distingue, attraverso uno switch-case(flag), l'azione richiesta ed invoca così la funzione handler opportuna.

5.1. Schema del db:



Relazioni: *carrello-utente* -> 1 a 1, *inserimento-carrello* -> N a N (molti a molti), *prodotto-inserimento* -> N a N, *acquisto-prodotto* -> N a N, *utente-acquisto* 1 a N (uno a molti).
Tutte le tabelle sono identificate da un attriburo "id" che si incrementa automaticamente ad ogni inserimento di una nuova tupla.
Le tabelle del database sono:

- *utente:* contiene le sue informazioni e un attributo "isAdmin" volto a distinguere il suo ruolo;
- *carrello:* ogni suo record corrisponde all'istanza di un determinato utente;
- *prodotto:* contiene tutte le informazioni sui prodotti mostrati nel sito;
- *acquisto:* contiene, per ogni utente, tutti gli acquisti da lui effettuati;
- *inserimento:* ogni suo record viene creato e quindi corrisponde all'azione di inserimento di un prodotto all'interno del carrello da parte dell'utente.

5.2. Descrizione delle funzioni remote: tutte le richieste ajax al server sono effettuate con metodo POST e laddove necessario vengono inviati dati tramite JSON.
Le funzioni remote sono:
1) *ModifyP.php, CartOp.php:* prende come parametro dal client un flag, come già descritto in precedenza tale meccanismo serve per distinguere le operazioni e in base a quest'ultime vengono presi ulteriori dati come input e rilasciati dati diversi in output:

A) Flag "INFOP" (ModifyP.php): in questo caso il server come dato vuole l'id del prodotto, che servirà a richiedere al database, tramite query SQL, le informazioni riguardanti il suddetto, restituite poi come output;
B) Flag "MODIFYPRODUCT" (ModifyP.php): in questo caso il server come dati vuole l'id del prodotto e le informazioni di esso modificate, a questo punto si procede con l'update nel database delle informazioni modificate, tramite query SQL, del prodotto selezionato;
C) Flag "COUNT" (CartOp.php): in questo caso il server come dato vuole l'id dell'utente a cui appartiene la sessione corrente, che servirà a richiedere al database, tramite query SQL, la lista dei prodotti da lui inseriti nel carrello,

restituita in output insieme all'id dell'utente e al numero dei prodotti (ricavato tramite il conteggio delle tuple estrapolate);

D) Flag "ADD"(CartOp.php): in questo caso il server come dati vuole l'id dell'utente a cui appartiene la sessione corrente e l'id del prodotto, in questo modo inseriamo all'interno del carrello dell'utente in questione, tramite query SQL, il prodotto da lui selezionato;

E) Flag "REMOVE"(CartOp.php): in questo caso il server come dati vuole l'id dell'utente a cui appartiene la sessione corrente, l'id e l'ora di inserimento del prodotto, si effettua così l'operazione di delete di quest'ultimo dal carrello dell'utente in questione, tramite query SQL;

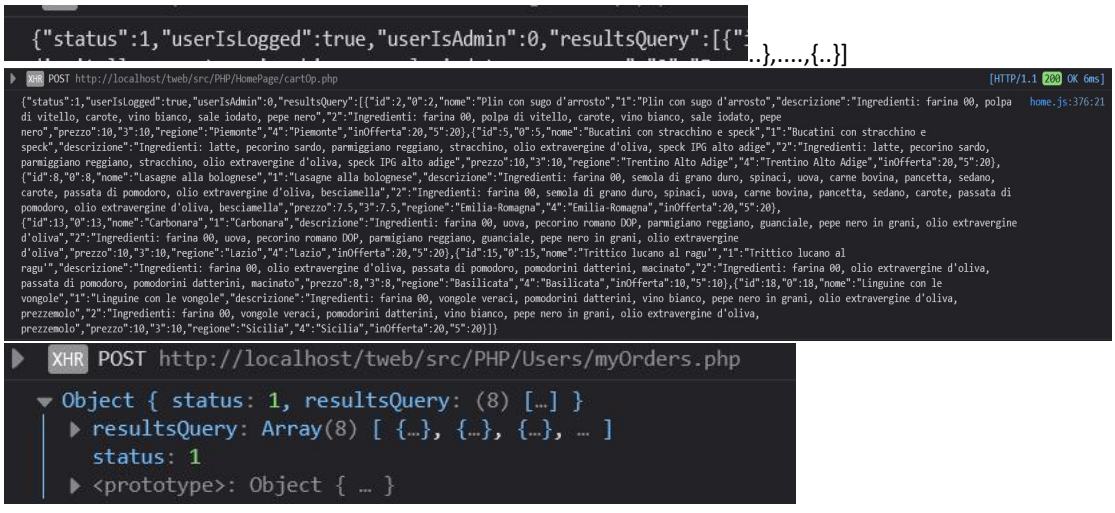
F) Flag "BUY"(CartOp.php): in questo caso il server come dati vuole l'id dell'utente a cui appartiene la sessione corrente, la lista di id dei prodotti inseriti nel carrello e l'indirizzo scritto all'interno del form dedicato per l'acquisto, a questo punto registriamo nella tabella degli acquisti, tramite query SQL, la lista di prodotti ricevuta come input per il suddetto utente all'indirizzo indicato.

2) *login.php:* prende come parametri del client l'email e la password dell'utente, in modo tale da verificare se sono stati inseriti dei valori corretti, tramite una query SQL di selezione sulla tabella utente;

3) *logout.php:* non richiede nessun parametro, viene distrutta la sessione corrente e viene infine restituita una risposta di successo;

4) *register.php:* prende come parametri del client tutti i dati che l'utente inserisce all'interno del form dedicato alla registrazione, tali valori serviranno per inserire l'utente all'interno della tabella utente, solo se esso non è stato inserito già precedentemente, e creare l'entry del suo carrello;

5) *home.php, myOrders.php e products.php:* non richiedono alcun parametro, vengono utilizzati però dati di sessione. L'output in caso di successo è un oggetto JSON così strutturato:



"resultsQuery" contiene l'insieme di tuple restituito della query di selezione, ogni riga corrisponde ad un prodotto.

6) *dateMyAccount.php:* non richiede nessun parametro, opera e restituisce i dati contenuti nell'array di sessione inizializzato al momento del login.