Trekk

Caricasulo Francesco

Matr. 944173

Tema del sito:

Il Tema di Trekk è quello di permettere a chiunque si avvicini al Trekking di poter avere accesso tramite una ricerca per regione o provincia, ad una lista di percorsi con varie informazioni e dettagli, e di poter stilare una lista con tutti i percorsi attraversati e che sono piaciuti.

Sezioni Principali:

Il progetto è suddiviso in 3 macro-sezioni:

- L'inizializzazione del sito tramite la pagina ***top.html*** che contiene il titolo del sito, e i richiami alle pagine css, php e js.

- La pagina principale ***index.php***, che corrisponde alla home iniziale del programma, infatti presenta i primi tre bottoni del banner e le scritte di benvenuto nella home.

- L'ultima sezione è la pagina ***index.js***, che contiene tutto il codice che permette di cambiare la visualizzazione della pagina in base alle azioni che l'utente compie.

Funzionalità:

Ciò che accoglie l'utente la prima volta che viene aperto il sito è la home, che contiene una frase di apertura, un'immagine e un banner con tre pulsanti con i quali è possibile interagire.

- Il pulsante *Registrazione*, una volta premuto, cambierà il contenuto della pagina aggiungendo una tabella tramite la quale potremmo inserire i dati per registrarci al sito: *Nome, Cognome, Username, Password* e se si è un *Gestore* o meno (Il Gestore è un utente che ha delle capacità in più sul sito, come se fosse una sorta di admin, oltre a poter ricercare e visualizzare percorsi, potrà anche aggiungerne di nuovi in base alla propria esperienza). Subito dopo aver premuto il pulsante registrati, il programma verificherà e tutti i campi siano pieni e invierà una richiesta ajax di tipo "POST" alla pagina *getData.php*, che ricevendo i dati si interfaccerà col database registrandoli. La pagina ritornerà un file json pieno in caso di successo e vuoto altrimenti. In base a ciò che sarà ritornato la funzione *successRegister* aggiungerà un avviso alla schermata.

- Il pulsante *Registra Percorso*, permetterà all'utente di tipo "Gestore", dopo aver fatto il login, di poter aggiungere nuovi percorsi al database attraverso la compilazione di una tabella che richiederà di aggiungere le seguenti informazioni: *Regione, Provincia, Partenza da, Arrivo a, Lunghezza* (Espressa in km) e *Difficoltà* del percorso. Subito dopo aver schiacciato il pulsante per registrare i dati ed eseguito i controlli sui campi, si invierà una richiesta ajax di tipo "GET" alla pagina *getData.php*

che registrerà il nuovo percorso sul database, ritornando una json pieno in caso di successo o vuoto altrimenti. Successivamente la funzione chiamata sarà sempre *successRegister*, che è stata differenziata in base a ciò che contiene il json cambiando il tipo di output se verrà registrato un utente o un percorso.

- Il pulsante *Accedi*, una volta premuto, cambierà il contenuto della pagina facendo comparire una tabella con richiesta di *Username, Password* e un pulsante di accesso. Una volta schiacciato il pulsante, prima verrà verificato che entrambi i campi siano stati compilati:
  -In caso di fallimento si illuminerà la casella alla quale mancano i campi;
  -In caso di successo verrà inviata una richiesta ajax di tipo "POST" alla pagina *getData.php* che si interfaccerà col database. La pagina php ritornerà un file json contenente l'Username e se l'account è un Gestore, nel caso in cui il profilo sia effettivamente presente sul database, altrimenti tornerà un file json vuoto. Successivamente alla richiesta ajax si passerà alla funzione *successLogin()* che gestirà la visualizzazione della pagina, cambiandola in caso di json pieno o aggiungendo una schermata di errore in caso di json vuoto.

- Subito dopo aver fatto l'accesso, sulla destra del banner, accanto al nome utente, ci sarà il tasto di *Logout*. Una volta premuto il tasto, apparirà la frase "Sei sicuro di voler uscire?", con due pulsanti sottostanti: "Sì" e "No". In caso di risposta negativa, si ritornerà alla schermata precedente, viceversa il sito invierà una richiesta ajax di tipo "GET" a *getData.php* che chiuderà la sessione, successivamente poi si ritornerà alla home del sito.

- Il pulsante *Informazioni* invece invierà una richiesta ajax di tipo "GET" alla cartella json per ricevere il file json2.json che contiene delle informazioni sul sito, per poi visualizzarle tramite la funzione successJson().

- Il pulsante *Percorsi Piaciuti* invia una richiesta ajax di tipo "GET" passando come parametro l'username dell'utente, alla pagina *getData.php* che verificherà nel database tutti gli Id dei percorsi connessi con l'utente, ritornandoli in file json. Successivamente si passerà alla funzione *visualizzaLiked()* che invierà una seconda richiesta ajax di tipo "GET" passando come parametro gli id dei percorsi trovati, in caso di successo metterà a schermo tutti i percorsi con le relative informazioni a cui l'utente ha messo like, oppure aggiungerà un testo in caso l'utente non abbia percorsi piaciuti.

- La ricerca vera e propria invece, viene eseguita tramite una barra che è presente sia nella home subito dopo aver eseguito il login, ma anche quando si entra nella sezione dei percorsi piaciuti o mentre si visualizzano già i risultati di una ricerca precedente. Una volta compilata la barra con la regione o la provincia dei percorsi cercati e premuto il pulsante accanto (Che controllerà se la barra sia stata effettivamente stata compilata), verrà inviata una richiesta ajax di tipo "GET" a *getData.php*, con il valore inserito all'interno della barra di ricerca. In caso si successo si riceverà un json contenente tutti i percorsi trovati nel database, in caso di fallimento verrà ritornato un json vuoto. Successivamente la pagina visualizzerà i percorsi, con le proprie specifiche, uno sotto l'altro, permettendo anche all'utente di aggiungere dei "mi piace".

- Una volta che viene premuto il tasto like, il programma invierà una richiesta di tipo "GET" a *getData.php* contenente l'username e i valori del percorso, la pagina chiamata prima ricercherà l'id del percorso

all'interno del database e successivamente lo collegherà col nome utente all'interno nella tabella "*likeutentepercorso*". Il numero di like del percorso verrà aggiornato e anche il valore del tasto verrà cambiato, passando da "like" a "dislike" che compirà l'operazione inversa una volta premuto.

Caratteristiche:

Usabilità: Ogni volta che l'utente esegue un'azione il programma fornisce a schermo un output differente. Durante la registrazione, subito dopo aver inserito correttamente tutti i dati il programma aggiungerà una sezione sottostante avvisando l'utente della buona riuscita dell'operazione, viceversa verrà avvisato in caso di mal funzionamento oppure di username già presente nel database. Verranno eseguite delle operazioni simili durante il login, o l'inserimento di un nuovo percorso, l'utente verrà avvisato anche in caso di nessun risultato riguardate il percorso da lui cercato, oppure se ancora non c'è nessun percorso nella propria lista di percorsi piaciuti.

Interazione/Animazione: Ogni pulsante sul banner del sito è animato, ogniqualvolta che l'utente posiziona sopra il proprio mouse il tasto si illumina evidenziandosi. Per quanto riguarda l'animazione, la sezione con i percorsi piaciuti viene aggiornata con ogni percorso al quale l'utente ha aggiunto/tolto un like.

Sessioni: La sessione dell'utente viene avviata quando si apre il sito, ma chiusa quando si effettua il Logout. In caso di aggiornamento della pagina mentre si è già loggati, la sessione rimarrà aperta tornando così alla home successiva al login. Quando la pagina viene aperta, il programma invia una richiesta ajax di tipo "GET" per verificare che effettivamente sia una nuova apertura della pagina, e non solo un aggiornamento della stessa. (La sessione ha una durata limite di inattività di 250 secondi).

Interrogazione del database: Il database viene interrogato mediante la pagina *getData.php*, quasi ogni volta che l'utente effettua un'azione sul sito, aggiungendo i propri dati in fase di registrazione, ricercandoli in fase di login, ricercando un percorso tramite Provincia o Regione, oppure aggiungendone uno nuovo nel caso in cui abbia la possibilità di farlo, anche quando si passa nella sezione dei percorsi piaciuti.

Validazione dati input: I dati vengono verificati prima di inviare la richiesta al database, intanto si controlla che tutti i campi siano effettivamente stati riempiti, e viene anche controllato che il dato inserito sia accettabile, cioè se il programma si aspetta una striga e invece vengono inseriti dei numeri o viceversa.

Sicurezza: Il codice riguardante le pagine visualizzate è scritto in javascript ciò evita possibili HTMLinjection; le richieste al database vengono effettuate sempre preparando prima la query e successivamente sostituendo i segnaposti con i valori effettivi prima di eseguirla, prevenendo così possibili problemi con l'SQLinjection lato server, mentre lato client, valido l'input inserito mediante la funzione "*isValidInput(input)*" che verifica effettivamente la validità del testo inserito. Per quando riguarda gli attacchi di tipo XSS, i dati sensibili vengono codificati tramite codifica md5; quindi, se qualcuno dovesse riuscire ad ottenere il dato, non sarebbe in grado di decifrarlo.

Presentazione: Il sito si presenta con una home abbastanza neutra, sulla destra si trova un'immagine di un percorso montano, che rimane per tutto il tempo in cui il sito è visualizzato mentre sulla sinistra è presente il testo del sito,

che presenterà poi ogni risposta all'interazione dell'utente. Tutto ciò è sovrastato da un banner che inizialmente presenta tre bottoni, ma che ne aggiungerà o rimuoverà altri in base agli spostamenti dell'utente.

Front-End:

- Il Front-End è organizzato secondo una separazione ben distinta, tutta la parte di stile è gestita mediante la pagina *index.css*, in cui sono inseriti tutti i dettagli riguardati l'organizzazione della pagina. Il contenuto della home iniziale è presente nella pagina *index.php*, tutto il resto viene gestito dalla pagina *index.js* che va a cambiare il contenuto dinamicamente, in base a ciò che l'utente compie o con cui interagisce. La pagina *top.html* invece è solo la struttura iniziale che serve per richiamare i vari script delle altre pagine.

- Per gestire la presentazione del sito in cross-platform, ho organizzato gli elementi definendone le dimensioni tramite valori percentuali, permettendo così un adattamento del contenuto in base alla grandezza della pagina.

- Il Progetto è scandito in varie cartelle all'interno del sistema, nella directory principale ci sono le 2 pagine php, la prima è "*index.php*" che è la home del sito, mentre l'altra "*getData.php*" è la pagina che gestisce l'interazione con il database; la pagina *top.html* serve per richiamare i vari script e contiene il titolo del sito e poi "*sol.json*" che è un file json temporaneo utilizzato per il ritorno dei vari dati ottenuti dal database. Il file "*index.css*" è contenuto nella cartella "css", così come le immagini utilizzate nel sito sono contenute in una cartella apposita "img", anche lo script "*index.js*" è contenuto nella cartella "js" e il file "*info3.json*", contenente le informazioni del sito, si trova nella cartella "json".

Back-End e Comunicazione:

(Architettura generale classi/funzioni php)

- Tutte le comunicazioni con il database sono gestite mediante la pagina *getData.php*, che gestisce le varie istruzioni in base al flag che gli viene passato o dal dato che riceve con prima linea alla funzione "GET" o "POST", tutto ciò viene gestito da una scala di *if* che controllano ogni operazione:

  Controllo sessione già avviata -> `if(array_key_exists('ric', $_GET))`

  Controllo per il login -> `if (isset($_POST['user1']) && isset($_POST['pass1']))`

  Controllo registrazione utente -> `if (array_key_exists('nome', $_POST))`

  Controllo registrazione percorso -> `if (isset($_GET['region']))`

  Controllo per ricercare percorso -> `if (array_key_exists('find', $_GET))`

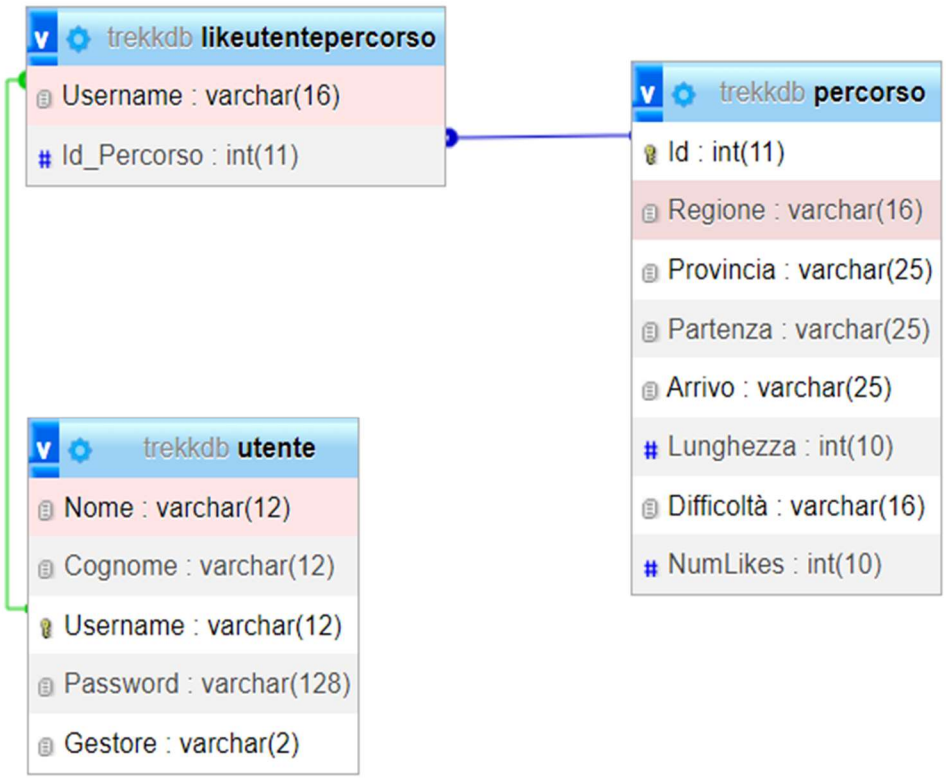
  Controllo per aggiungere/rimuovere like -> `if (array_key_exists('type', $_GET))`

  Controllo percorsi piaciuti dall'utente -> `if (isset($_GET["lkd"]))`

  Controllo per la chiusura della connessione -> `if (array_key_exists('tipo', $_GET))`

(Schema del DB)

- Il Database "*trekkdb*" è organizzato in 3 tabelle: la prima è la tabella "*likeutentepercorso*" che serve per collegare l'username di un utente (chiave primaria della tabella utente) con l'id di un percorso (chiave primaria della tabella percorso); la tabella "*percorso*" serve per la gestione dei percorsi presenti, presenta 8 campi: *l'id(auto-incrementante), regione, provincia, il luogo di partenza, il luogo d'arrivo, la lunghezza, la difficoltà e il numero di like*. La tabella "*utente*" invece serve per la gestione degli utenti iscritti, presenta 5 campi: *Nome, Cognome, Username, Password (codificata in md5) e Gestore.*



(Descrizione Funzioni Remote)

- La prima funzione remota richiamata da *index.js* è una funzione anonima quando la pagina viene caricata, che esegue una chiamata ajax di tipo "GET" all'url: "*getData.php*" passandogli come parametro un vettore contenente come campo il valore "ric", che verrà utilizzato per effettuare un controllo sulla sessione se sia stata già aperta o meno, in caso positivo verrà ritornato un file JSON temporaneo, di nome "sol", che può essere vuoto in caso di nessun riscontro, oppure strutturato:

  ```
  {
   "Username": username dell'utente già loggato;
   "Gestore": se l'utente è gestore o meno;
  }
  ```

  il json è pieno verrà richiamata la funzione *successLogin(json)* che strutturerà la pagina principale in maniera che venga visualizzata l'interfaccia successiva al login, altrimenti si vedrà la home principale del sito.

- La funzione *login()* effettua una chiamata remota di tipo "POST" all'url: "*getData.php*", passandogli come parametro l'username dell'utente e la password e ricevendo dalla funzione un file JSON temporaneo, di nome "sol", che può essere vuoto in caso di nessun riscontro, oppure strutturato:

  ```
  {
   "Username": username dell'utente;
   "Gestore": se l'utente è gestore o meno;
  }
  ```

  Verrà poi chiamata la funzione *successLogin(json)* che modificherà la pagina in base a ciò che sarà presente all'interno del file ricevuto come parametro

- La funzione *siClick()* effettua una chiamata ajax di tipo "GET" all'url:"*getData.php*" passandogli come parametro un array contenente il valore "esc" che servirà per gestire l'uscita. Questa istruzione non riceve nessun valore di ritorno, e in caso di successo richiama la funzione *homeClick()* che fa ritornare il sito alla home principale.

- La funzione *register1()* viene utilizzata per registrare un nuovo utente al database, essa effettua una chiamata ajax di tipo "POST" verso "*getData.php*" passandogli come parametro i valori inseriti dall'utente in fase di registrazione e ricevendo un file JSON temporaneo, di nome "sol", che può essere vuoto in caso di nessun riscontro, oppure strutturato:

  ```
  {
   "Username": "Successo1";
  }
  ```

  Verrà poi richiamata la funzione *successRegister(json)* che in base al valore contenuto nel json farà comparire un avviso di avvenuta registrazione o un errore.

- La funzione *register2()* ha la stessa dinamica di register1, solo che passerà come dati i campi riguardanti il percorso e il json di ritorno conterrà come Username "Successo2" anziché "Successo1". Anche la funzione richiamata in caso di successo sarà la stessa (*successRegister(json)*), che

aggiungerà la schermata di avviso in caso di successo nella registrazione o di errore.

- La funzione *cercaP()* viene utilizzata per ricercare i percorsi dall'utente, effettua una chiamata ajax di tipo "GET" verso "*getData.php*" passandogli come parametro il campo ricercato e ricevendo un file JSON temporaneo, di nome "sol", che può essere vuoto in caso di nessun riscontro, oppure strutturato:

  ```
  {
   "Regione": regione percorso;
   "Provincia: provincia percorso;
   "Partenza": luogo partenza del percorso;
   "Arrivo": luogo d'arrivo del percorso;
   "Lunghezza": lunghezza del percorso;
   "Difficoltà": difficoltà del percorso;
   "NumLikes": numero di like del percorso;
  }
  ```

  In caso di successo nella chiamata, verrà eseguita la funzione *visualizzaP(json)* che metterà a schermo tutti i percorsi trovati, tenendo conto se l'utente ha già messo like o meno ad ogni percorso (tramite una chiamata ajax interna di tipo "GET" verso "*getData.php*" e ricevendo un file json, che conterrà: {"Username": true/false;}) e in base a ciò, cambierà il tipo di pulsante visualizzato "like/dislike".

- La funzione *likeClick()* serve per permettere all'utente di aggiungere/rimuovere un like dai percorsi, questa funzione prima effettua una ricerca per verificare se si tratterà di un like o di un dislike, e successivamente effettua una chiamata ajax verso "*getData.php*", passandogli come parametro il tipo dell'operazione, la partenza del percorso, l'arrivo, la lunghezza(Usate per rintracciare l'id del percorso) e l'username dell'utente e ricevendo un file JSON temporaneo contenente il numero dei like aggiornati del percorso:

  ```
  {
  "Username": numero di like;
  }
  ```

  utilizzati poi, in caso di successo, da una funzione anonima per aggiornare il contatore.

- La funzione *likedP()* permette all'utente, quando viene richiamata, di visualizzare tutti i percorsi a cui ha messo like, eseguendo una chiamata ajax verso "*getData.php*", e ricevendo un file JSON temporaneo, di nome "sol", che può essere vuoto in caso di nessun riscontro, oppure strutturato come:

  ```
  {
   "Regione": regione percorso;
   "Provincia: provincia percorso;
   "Partenza": luogo partenza del percorso;
   "Arrivo": luogo d'arrivo del percorso;
   "Lunghezza": lunghezza del percorso;
   "Difficoltà": difficoltà del percorso;
   "NumLikes": numero di like del percorso;
  }
  ```

In caso di successo, verrà richiamata la funzione *visualizzaLiked(json)* che si occuperà di visualizzare tutti i percorsi presenti nel file, oppure si far comparire una frase di avviso in caso di nessun percorso presente.

- La funzione *infoClick()* compirà una chiamata ajax di tipo "GET" all'url locale "*json/info3.json*", ricevendo un file json organizzato:

```
"info":[
      {
      "title": titolo della pagina;
      "body": descrizione sull'utilità del sito;
      "btn": corpo per un bottone che compare dopo la chiamata;
      }
]
```

In caso di chiamata avvenuta con successo, verrà eseguita la funzione *successJson(json)* che si occuperà di formattare la pagina visualizzando i valori contenuti nel file json.

- La funzione *homeClick()* compirà una chiamata ajax di tipo "GET" all'url locale "*json/info3.json*", ricevendo un file json organizzato:

```
"home":[
      {
      "title": titolo della pagina;
      "body": Breve frase all'apertura della home;
      "btn1": Accedi;
      "btn2": Registrati
      "btn3": Informazioni;
      }
]
```

In caso di chiamata avvenuta con successo, verrà eseguita la funzione back(json) che si occuperà di formattare la pagina facendo ricomparire i valori della home.

Ogni chiamata ajax effettuata, in caso di mancata connessione con il database, e quindi errore, richiamerà la funzione *error()* che si occuperà di comunicare all'utente la mancata connessione con il database facendo comparire una scritta d'avviso.