

UNIVERSITA' di TORINO

Progetto di Tweb 2020/2021

Demis Mazzotta

corso B

MATR 814574

Tema del sito:

Style Your Life

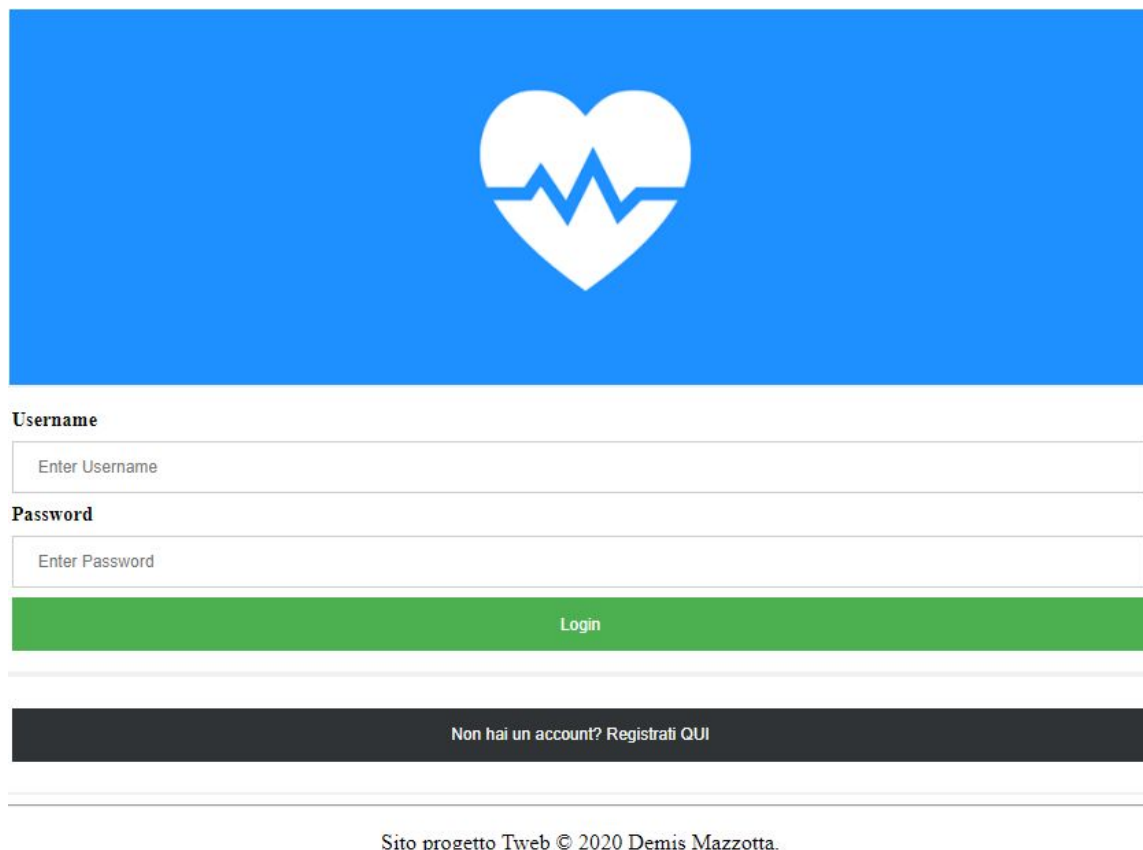
Il sito nel suo piccolo permette di tenere il profilo fisico di un utente, permettendogli di calcolare Indice di massa corporea, fabbisogno energetico e, metabolismo basale.

Tali dati sono poi automaticamente salvati per permettere all'utente di vedere i suoi progressi fisici e i suoi sviluppi

Sezioni Principali:

- Pagina principale (index.php): rappresenta l'accesso al sito, con modulo di login e registrazione
 - Statistiche: pagina di recap, accessibile solo se loggati, permette la visualizzazione dei vari calcoli e risultati dei test eseguiti nel sito
 - Calcolo IMC: sezione per calcolo indice massa corporea con invio automatico dei dati
 - Calcolo metabolismo: sezione per calcolo metabolismo e fabbisogno energetico
 - Push UP Game: gioco interattivo d'intrattenimento, dove si fa sgobbare un povero studente sotto esame
 - I tuoi Dati: pagina che permette inserimento iniziale dei dati e visualizzazione di questi ultimi
-

Funzionalità: index.php



The screenshot shows a login form with a blue header containing a white heart icon with a blue ECG line. Below the header, there are two input fields: 'Username' and 'Password', both with placeholder text 'Enter Username' and 'Enter Password' respectively. A green 'Login' button is positioned below the password field. At the bottom, a dark gray bar contains the text 'Non hai un account? Registrati QUI'. The footer of the page reads 'Sito progetto Tweb © 2020 Demis Mazzotta.'

Il form principale inizialmente era in comunicazione sincrona php, senza ricorrere ad ajax, successivamente e' stato convertito anche esso in ajax

L'utente inserisce username e password:

- se l'utente non esiste, compare un messaggio di utente non presente e viene reindirizzato sul form di registrazione.

- se l'utente esiste ma ha inserito una password errata, compare un messaggio di errore e si rimane sulla stessa pagina

- entrambi gli input sono impostati required(client) e controllati tramite la funzione `strip_tags`(server) per bloccare attacchi XSS

register.php



Nome:

Password

E-mail:

REGISTRATI

Sito progetto Tweb © 2020 Demis Mazzotta.

Form di registrazione semplice, Nome in chiaro, password oscurata. Anche qui gli input sono impostati required senza particolari filtri.

Anche qui inizialmente l'approccio e' stato sincro con php e successivamente convertito in ajax, i dati vengono alla pagina signup.php che spieghero' in seguito, e in base alle risposte chiamo funzioni di callback

Lato server, la password e' criptata con md5 e i valori sono passati nelle query attraverso la funzione `BindValue` e non `quote`, dato che svolge una funzione simile concedendo solamente i caratteri '%' e '_' anche qui sono applicati i medesimi controlli con `strip_tags(server)`

statistiche.php

In questa pagina puoi vedere i tuoi progressi e l'andamento del tuo lavoro

questi sono i tuoi progressi demis2

data	peso0	imc	metabolismo	pesoX
2021-01-17	50	17.3	3251.17	75
2021-01-18	50	17.3	3251.17	75

Stampa i Risultati

Condividi i Risultati

Tale pagina non è accessibile se l'utente non è loggato.

I dati ottenuti sono risultato di varie query eseguite con chiamata asincrona AJAX verso la pagina "sendValue.php"

precisamente: `getID(nome,pdo)` = query che si occupa di trovare l'id dell'utente loggato usando il suo username (prelevandolo dalla `$_SESSION`) come campo di ricerca nella table users del Database.

`querystats(id,pdo)` = query che ritorna tutti i dati associati all'utente in formato json riga per riga. Successivamente ho aggiunto un form select, con il quale si possono ordinare i risultati

```
{idUtente: "27", data: "2021-01-17", peso0: "50", imc: "17.3", fabbisogno: "3251.17", pesoX: "75"}
```

<pre>data: "2021-01-18" fabbisogno: 3251.17" idUtente: "27" imc: "17.3" peso0: "50" pesoX: "75"</pre>	<p>lato Javascript invece come funzione di callback ho un ciclo per la lunghezza del json. quindi creo un div per ogni elemento ricevuto e gli assegno il valore json corrispondente. successivamente appendo il nuovo div alla griglia</p>
---	---

I due button invece sono associati a due funzioni javascript, rispettivamente: "Stampa i risultati" -> esegue una semplice print della pagina. "Condividi i risultati" -> sfrutta la libreria html2canvas per eseguire uno screen della griglia e scaricarlo come immagine dal browser

calcoloIMC.php e metabolismo basale

Entrambe le pagine presentano un form di immissione dati che processano in locale i dati e inviano successivamente tramite ajax i risultati al server (andranno a popolare la pagina statistiche.php)

caso IMC : successivamente eseguo funzioni callback in cui setto visibile i risultati e chiamo la funzione `newDiv` che crea il div con i button di stampa e condividi visti precedentemente

caso Fabbisogno: simile ma senza i button di condivisione

Lato server, ho delle funzioni con query che si occupano di trovare l'id dell'utente tramite l'username della session, successivamente fanno un check della data, se la data dell'ultima entry dell'utente e' diversa da quella generata durante i calcoli, allora faccio una nuova INSERT dei dati con la rispettiva data (`insertStatistiche`)

Altrimenti UPDATE della entry relativa alla data. (`updateStatistiche`)
Ho scelto questo approccio perche' magari l'utente esegue più volte il test per provare, quindi nel recap tengo conto solo dell'ultimo test eseguito.

pushUP GAME.php

la pagina come anticipato e' un semplice giochino di intrattenimento e non comunica in nessun modo con il server

ha un timer che al via sblocca la funzione di click `changeImage` :
la quale, finche' il timer non e' a 0, modifica tramite un semplice if, la source dell'immagine e incrementa il counter.
tale counter viene mostrato e refreshato volta per volta, modificando lo span tramite la proprietà `innerText`.

progressi.php

Questa pagina si presenta in due alternative, ed e' abbastanza fatta male, inizialmente come tutte le altre pagine era in comunicazione php sincrona, successivamente convertita a meta' in ajax.

Si occupa di inviare le generalita' dell'utente appena registrato (il quale non ha ancora inserito nessun dato), avrei potuto anche aggiungere una funzione di modifica di tali dati, ma essendo che non hanno un reale scopo, ne faccio a meno.

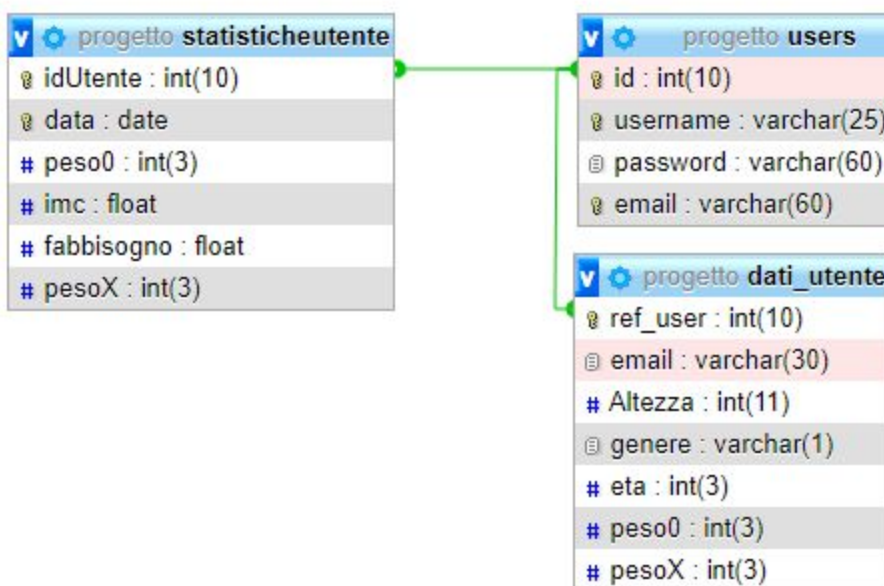
La pagina quindi inizialmente fa un check in php sui dati nella table **dati_utente**, se sono presenti mostra semplicemente i dati, altrimenti mostra un form.

l'invio dei dati funziona quindi con ajax verso la pagina

"ajaxsend.php" e successivamente effettua un refresh della page.

Caratteristiche :

Server Database:



General info:

table **users**: ho impostato l'id con auto incremento.

la password come già detto e' criptata.

le tabelle sono referenziate come mostrato nello schema con vincoli on cascade, quindi alla cancellazione dell'utente

-alla registrazione dell'utente creo delle nuove entry nelle table "statisticheutente" e "dati_utente" impostando la data e l'id

table **statisticheutente**: contiene i dati passati tramite le call ajax dalle pagine per l' IMC e il metabolismo.

conterra' una entry per ogni data diversa in cui si fanno i calcoli

table **dati_utente**: contiene i dati personali iniziali, quindi altezza, eta' ecc ecc, non hanno uno scopo vero per il sito, ma sono stati introdotti piu per una funzione di generalita', tipicamente ogni sito del genere profila i dati degli utenti.

Caratteristiche :

Comunicazioni: Server (Connect.php)

Per quanto riguarda la connessione iniziale, dopo la definizione delle variabili di accesso al server, effettuo un piccolo controllo sulla session con un ifset, nel caso in cui non fosse presente la attivo.

La connessione avviene successivamente con l'oggetto PDO gestito con un try-catch.

Per le pagine che hanno bisogno di un login per funzionare ho usato il metodo visto a lezione [LoggedIN](#) , con il quale controllo se e' presente l'username nell'array della \$_SESSION, ed eventualmente effettuo un redirect sulla page del login (index.php)

Server (login.php)

Riguardo al login: controllo se i valori username e password non sono vuoti e successivamente uso la funzione `strip_tags`. Controllo quindi tali valori con i risultati della query sulla table **users**, se i dati corrispondono allora assegno sia l'username e la mail alla `$_SESSION` e assegno un messaggio di controllo alla `$_SESSION["flash"]`, successivamente ritorno ad ajax un array json con un valore di risultato (`res`) e la pagina da richiamare (`page`).
come funzione di callback success quindi controllo il `res` ed in base a ciò reindirizzo la pagina usando il valore di 'page' ove necessario, tipo quando l'utente non esiste nel DB
mentre

```
{res: 0, page: "index.php", text: "utente e/o password errati"}
  page: "index.php"
  res: 0
  text: "utente e/o password errati"
```

Server (submit.php)

Riguardo alla registrazione : metodo simile, sempre con ajax e funzione di callback
passo i valori, li controllo tramite delle regex con la funzione `preg_match` e `strip_tags`
Eseguo una query che fa un count degli username con il nome passato, se ho un valore `>0`, vuol dire che l'utente e' gia presente quindi ritorno il solito array con il `res`, la `page`, e il `text`. altrimenti creo una entry nella table **user** e successivamente anche nelle altre table **statisticheutente** e **dati_utente** tramite le funzioni `createEntrydati_Utente` e `createEntryStatistiche`, successivamente reindirizzo sulla pagina index per il login.
