Documentația aplicației - Cabinet Psihologic

Gavriluț Adriana

Funcționalități:

1. Pagina Home:

- autentificare
- chat, care te trimite la login (psihologii și pacienții pot avea conversații)
- afișarea doctorilor înregistrați
- afișarea testimonialelor înregistrate
- afișarea celor mai recente 4 articole

1.1 Pagina Articole

- vizualizarea tuturor articolelor
- accesarea unui articol pentru a-l vedea în întregime
- search după titlul articolelor

2. Admin - când este logat

- adăugarea, editarea și ștergerea psihologilor
- adăugarea, editarea și ștergerea articolelor
- adăugarea, editarea și ștergerea testimonialelor
- toate trei au opțiunea de a încărca o imagine

3. Medic - când este logat

- adăugarea, editarea și ștergerea pacienților; pentru fiecare pacient:
  - vizualizarea testelor (cu întrebări și răspunsurile date de pacient)
  - vizualizarea istoricului (cu datele introduse de pacient)
- adăugarea, editarea și ștergerea testelor; pentru fiecare test:
  - adăugarea, editarea și ștergerea întrebărilor
- chat: începerea unei conversații cu oricare dintre pacienți

4. Pacient - când este logat

- vizualizarea testelor
  - accesarea unui test și răspunderea la întrebări
- completarea istoricului medical
- chat: începerea unei conversații cu psihologul său

5. Autentificare

- trimitere mail - resetare parola
- trimitere mail - activare cont

Template link:

- https://themewagon.com/blog/responsive-templates-for-medical-websites/
- https://bootstrapmade.com/demo/FlexStart/
- https://startbootstrap.com/theme/sb-admin-2

Tehnologii folosite:

- PHP (limbaj de programare)
- Twig (motor de șablon)
- Mysqli (driver de bază de date)
- HTML (limbaj de marcare)
- CSS (standard pentru formatarea elementelor unui document HTML.)
- JS (limbaj de programare JavaScript)
- jQuery (platformă de dezvoltare JavaScript)
- AJAX (tehnică de programare pentru crearea de aplicații web interactive)
- Boostrap (framework CSS)
- Font Awesome (librărie de icoane)
- Gijgo (set de controale javascript)
- Quill (editor WYSIWYG)
- Mailjet (sistem de livrare și urmărire a e-mailurilor bazat pe cloud)

Versiuni:

- PHP: 7.3.2
- Server DB: mysql:5.7
- Boostrap: v5.3 (pentru home), v4.6 (pentru administrare)

Modelul arhitectural Model-view-controller:

- Fiecare entitate (utilizator, psiholog, pacient, test, întrebare, răspuns, articol, testimonial, chat) este reprezentată de câte o clasă în Php. Clasele sunt formate din câmpurile specifice, un constructor și metode. Rolul acestor metode este de a lucra cu informațiile din baza de date (prin Mysqli): obținere, editare, inserare, ștergere. Acestea formează partea de *model*.
- Aceste date sunt cerute și obținute particularizat de către așa numitele *controllere*, fișiere Php care le redau în șabloanele respective. Componenta *controller* acționează ca intermediar între *model* și *view* și gestionează interacțiunea utilizatorului cu aplicația. Fișierele PHP, numite *controlere*, conțin logica de control a aplicației. *Controlerele* primesc cereri de la utilizator prin intermediul rutelor definite și apelurile HTTP (GET, POST) și decid cum să răspundă la aceste cereri. Controlerele obțin datele necesare din *model* și le trimit către *view* pentru a fi afișate utilizatorului.
- S-a ales un motor de șablon pentru limbajul de programare PHP, Twig, astfel că în fișierele din *views* (.html.twig) sunt integrate datele necesare împreună cu structura paginii Web dată de codul HTML, cu stilizarea din link-urile CSS și funcționalitățile date de script-urile JavaScript și sunt încărcate paginile în browser. Componenta *view* este responsabilă de prezentarea datelor utilizatorului și de interfața grafică a aplicației.

- MVC promovează separarea clară a responsabilităților între aceste trei componente, permițând o dezvoltare mai structurată, modulară și ușor de întreținut a aplicației. *Modelul* asigură logica și starea datelor, *View-ul* se ocupă de prezentare, iar *Controller-ul* coordonează interacțiunea între cele două și răspunde cererilor utilizatorului.

Despre baza de date:

- 11 tabele
- tabela Utilizator: conține informații (nume, email, parola, roluri și coloane despre statutul contului) despre cele trei tipuri de utilizatori admin, psiholog, pacient; de acest tabel sunt legate tabelele: Medic, Pacient, Conversație și Mesaj
- tabela Medic: conține informațiile personale despre fiecare psiholog; e legată de tabela Pacient, Test și Întrebare
- tabela Pacient: conține informațiile personale despre fiecare pacient; e legată de tabelele: PacientTest și Răspuns
- tabela PacientTest: face legătura dintre un pacient și un test

Descriere funcții deosebite:

1. Dropzone pentru chat

a. S-a inclus librăria Dropzone.js
b. Dropzone.js este o librărie JavaScript care facilitează încărcarea simplă și elegantă a fișierelor în aplicațiile web. Cu Dropzone.js, utilizatorii pot trage și plasa fișierele sau pot da clic pentru a le selecta și încărca într-o zonă specificată.
c. Funcționalitatea Dropzone.js se bazează pe integrarea acestei librării în codul HTML și în JavaScript-ul proiectului. Zona de încărcare, definită cu ajutorul clasei CSS "dropzone", este afișată utilizatorului, iar acesta poate trage și plasa fișierele în această zonă sau poate face clic pentru a deschide dialogul de selecție a fișierelor. După ce fișierele sunt selectate sau trase în zona de încărcare, Dropzone.js preia controlul și gestionează procesul de încărcare. Utilizatorul poate vizualiza progresul, succesul sau eșecul încărcării.

2. Procesul de creare psiholog

a. În fișierul "new.php" avem următorii pași:
b. Verificăm autentificarea utilizatorului și dacă acesta are drepturi de administrator. Dacă utilizatorul nu este autentificat sau nu are drepturi de administrator, acesta este redirecționat către pagina de autentificare.
c. Dacă metoda de cerere HTTP este "POST", continuăm procesul.
d. Verificăm și salvăm imaginea psihologului în directorul corespunzător. Imaginea este încărcată în funcție de câmpul "imagine" din formularul de creare.
e. Apelăm metoda "create" a clasei "Medic" pentru a crea un nou obiect "Medic" în baza de date. Transmitem informațiile necesare pentru creare: email, nume, specialitate, data de naștere, telefon, adresă și calea către imaginea psihologului. Dacă crearea psihologului este realizată cu succes, utilizatorul este redirecționat către pagina de listare a psihologilor.
f. În caz contrar, afișăm template-ul "new.html.twig" cu erorile corespunzătoare.

g. În clasa Medic: metoda statică "create" realizează procesul de creare a unui psiholog în baza de date. Verificăm și validăm datele primite, inclusiv existența și unicitatea adresei de email. Dacă totul este în regulă, salvăm psihologul în baza de date, creăm un utilizator asociat și trimitem o invitație prin email pentru activarea contului.

3. Salvarea anamnezei introdusă de pacient

a. În fișierul pacient/index.html.twig pentru secțiunea de istoric, avem un formular înconjurat de un element div cu id-ul "istoric". Acest formular are un atribut data-pacient-id care stochează id-ul pacientului curent.
b. În fișierul "istoric.php", avem următorii pași:
    i. Verificăm autentificarea utilizatorului și dacă acesta este pacient. Dacă utilizatorul nu este autentificat sau nu are drepturi de pacient, acesta este redirecționat către pagina de autentificare.
    ii. Obținem obiectul pacientului curent utilizând metoda statică "getForUserId" din clasa "Pacient".
    iii. Apelăm metoda statică "updateHistory" a clasei "Pacient" pentru a actualiza istoricul medical al pacientului. Transmitem ID-ul pacientului, numele câmpului și valoarea acestuia.
    iv. În cazul în care actualizarea este realizată cu succes, returnăm "true", iar în caz contrar returnăm erorile corespunzătoare.
c. În clasa "Pacient", avem metoda statică "updateHistory" care realizează actualizarea istoricului medical al pacientului:
    i. Creăm conexiunea la baza de date.
    ii. Construim interogarea SQL pentru a actualiza câmpul specificat din tabelul "Pacient" cu noua valoare.
    iii. Executăm interogarea și verificăm rezultatul. În caz de succes, returnăm "true", iar în caz contrar returnăm erorile corespunzătoare.
    iv. Închidem conexiunea cu baza de date.
d. În fișierul "istoric.js", avem următorii pași:
    i. Folosim funcția jQuery $(document).ready() pentru a asigura că codul este executat după ce pagina s-a încărcat complet.
    ii. Utilizăm evenimentul "keyup" și "paste" pentru a detecta schimbările în câmpurile de intrare cu clasa "js-ancestor".
    iii. Pentru fiecare schimbare, efectuăm o cerere AJAX către fișierul "istoric.php" folosind metoda "POST".
    iv. Transmiterea datelor se face prin obiectul data, unde coloana reprezintă numele câmpului și valoare reprezintă noua valoare introdusă de pacient.
e. Acești pași și fișierele menționate permit salvarea istoricului medical introdus de pacient în aplicație. Modificările introduse în câmpurile de intrare sunt trimise în mod asincron prin cereri AJAX către server, unde se realizează actualizarea în baza de date. Astfel, informațiile introduse de pacient sunt salvate și actualizate în timp real.